MASTER EN SCIENCES MATHÉMATIQUES

Représentation des Résultats de l'Analyse Structurale en Paysages 3D

Roland, Francois

Award date:
2001

FUNDP
Faculté des Sciences
Département de Mathématique

Rempart de la Vierge, 8
B-5000 Namur Belgique

Représentation des résultats de l'analyse structurale en paysages 3D

Promoteur : RÉMON Marcel

Mémoire présenté pour l'obtention
du grade de
Licencié en Sciences Mathématiques
par

ROLAND François

Année académique 2000-2001

Je tiens tout d'abord à remercier Marcel Rémon, mon promoteur de mémoire, qui m'a permis par la réalisation de ce travail de découvrir et de prendre goût à la modélisation 3D, un domaine qui m'était inconnu. Je le remercie également pous ses conseils et sa disponibilité. Je tiens également à remercier Anne Wallemacq et Jean-Marie Jacques. En effet, j'ai pu grâce à eux m'ouvrir quelque peu à un autre domaine que celui des mathématiques et de la programmation. Collaborer au projet sur lequel ils travaillent depuis quelques années maintenant fut très passionnant. Je tiens aussi à remercier Fabien Bastin et Benoît Colson de m'avoir souvent apporté une aide précieuse pour résourdre certains problèmes. Enfin, je tiens à remercier mes parents, ma famille ainsi que ma petite amie pour leur soutien tout au long de mes études et particulièrement lors de la réalisation de ce mémoire.

Résumé

L'analyse structurale est une méthode d'analyse de contenu visant à faire émerger le sens d'un texte par la mise en évidence des interrelations entre les mots qui le composent. Depuis quelques années, A. Wallemacq, J.-M. Jacques et K. Bertels du département des Sciences Economiques, Sociales et de Gestion aux Facultés Universitaires Notre-Dame de la Paix à Namur travaillent sur un projet de visualisation sous forme de paysages 3D des résultats de ce type d'analyse. Ce mémoire se situe essentiellement dans la réalisation informatique de ce projet, à savoir l'écriture d'un programme générant des paysages 3D sur base des résultats de l'analyse structurale. Nous avons été amené dans le cadre de ce travail à utiliser un outil récent en matière de modélisation 3D : le langage *VRML* (*Virtual Reality Modeling Langage*).

Abstract

Structural analysis belongs to contents analysis methods and aims to show the sense of a text by bringing to the fore interrelations between the words of that text. For a few years, A. Wallemacq, J.-M. Jacques and K. Bertels, all members of the "*Département des Sciences Economiques, Sociales et de Gestion*" at the "*Facultés Universitaires Notre-Dame de la Paix*" in "*Namur*" work on a project of visualisation with 3D landscapes of the results of such analysis. This report takes place essentially in the realisation of the computer side of that project, more precisely in the writing of a program generating 3D landscapes from structural analysis results. During this work, we were induced to use a new tool relating to 3D modelisation : *VRML* (*Virtual Reality Modeling Langage*).

Table des matières

Introduction

De tout temps, l'homme s'est servi du langage visuel pour communiquer ou pour traduire ses émotions. Il n'y a rien d'étonnant à cela: le langage visuel est sans doute universel.

Ces dernières années, grâce à la puissance sans cesse croissante des ordinateurs et grâce aux développements considérables dans le domaine de l'informatique, les hommes se sont tournés vers ces nouvelles technologies dans le but de les associer au langage visuel pour renforcer davantage celui-ci. Le terme *graphisme informatique* désigne d'ailleurs cette alliance des techniques informatiques aux méthodes de communication au moyen du langage visuel (photographies, graphiques, ...).

Aujourd'hui, cette technologie visant à développer des interactions entre les machines et les sens humains est connue sous l'appellation de "Réalité Virtuelle". Ainsi, on ne se limite plus seulement à la vue. On envisage également les autres sens: ouïe, toucher, odorat.

L'analyse structurale[1] n'échappe pas à ce désir de vouloir s'appuyer sur le langage visuel et sur les technologies informatiques pour renforcer la pensée et la représentation (interprétation) de résultats.

Sous la direction de A. Wallemacq, J.-M. Jacques et K. Bertels, se développe depuis quelques années aux Facultés Universitaires Notre-Dame de la Paix à Namur le projet EVOQ. Il s'agit de la création d'un logiciel visant à fournir une visualisation sous forme de paysages en trois dimensions, des

1. Nous consacrons un chapitre à l'analyse structurale mais disons déjà qu'il s'agit d'une méthode d'analyse de contenu visant essentiellement à faire émerger le sens d'un texte par la mise en évidence des interrelations entre les mots du texte.

résultats de l'analyse structurale.

Ce mémoire porte sur la génération de réalités virtuelles (i.e. des scènes -paysages- 3D avec lesquelles l'observateur peut interagir) reflétant les résulats provenant de l'analyse stucturale appliquée à un texte quelconque. Un paysage devra ainsi évoquer l'essence même d'un texte selon les principes de l'analyse structurale. Notre travail se situe essentiellement dans la réalisation informatique de ce projet.

Nous avons divisé ce mémoire en six chapitres.

Le premier chapitre présente les notions de base de l'analyse structurale nécessaires à la compréhension du projet. Le deuxième chapitre aborde la modélisation graphique retenue pour représenter les différentes structures rencontrées en analyse structurale et présentées au chapitre 1. Le chapitre 3 est un chapitre plutôt mathématique. Nous développons certains aspects de géométrie en 3D qui seront utiles lors de la conception du programme. Nous consacrons un chapitre à cela pour ne pas surcharger le chapitre 4 qui est déjà assez conséquent. Ce quatrième chapitre est le plus technique: nous développons pas à pas le fonctionnement interne du programme. Le chapitre 5 explique comment se servir du programme et présente quelques résultats. Un chapitre de conclusion donne quelques perspectives pour un travail à venir.

Nous fournissons en annexe un guide d'utilisation du langage *VRML* (*Virtual Reality Modeling Langage*) utilisé pour visionner les réalités virtuelles générées par le programme. En effet, le *VRML* est un langage assez récent (1993) et nous avons trouvé bon de rédiger cette annexe pour que le lecteur puisse se familiariser rapidemment avec celui-ci. Nous joignons également en annexe le code du programme et les fichiers de définition des objets *VRML*.

Chapitre 1

Les grands principes de l'analyse structurale

Ce chapitre présente les idées de base de l'analyse structurale. Nous n'avons retenu que les concepts qui seront utiles lors de la modélisation dans un premier temps et de l'implémentation dans un second temps.

Nous faisons dans ce chapitre essentiellement référence à [10].

1.1 Introduction

Une définition satisfaisante de ce qu'est l'analyse structurale[1] pourrait être celle-ci : *l'analyse structurale fait partie des méthodes d'analyse de contenu qualifiées de sémantiques et de structurales.*

La figure[2] 1.1 devrait éclairer quelque peu cette définition.

Commentons quelque peu la figure 1.1. Les méthodes sémantiques se penchent avant tout sur le sens du discours; les méthodes logico-esthétiques et formelles s'intéressent quant à elles aux caractéristiques psycho-affectives, socio-affectives ou socio-linguistiques du locuteur révélées par les caractéristiques de son discours. Les méthodes logico-sémantiques ne s'en tiennent

1. La méthode a été initialement élaborée par J.P. Hiernaux [7] au départ de A.J. Greimas [6].

2. La classification apparaissant à cette figure est due à Mucchielli.

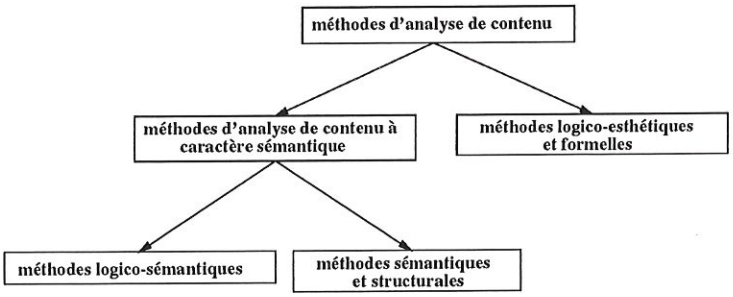


FIG. 1.1 – *Tentative de classification des méthodes d'analyse de contenu.*

qu'au contenu directement manifesté du discours. Enfin, les méthodes sémantiques et structurales cherchent à atteindre par une analyse au second degré le sens implicite du discours.

L'analyse structurale est une méthode d'analyse sémantique dans la mesure où on s'intéresse au sens du discours : on vise à identifier les modèles de pensée du locuteur car ceux-ci structurent sa représentation, sa compréhension du monde. Autrement dit, l'objectif poursuivi est d'attribuer le sens que le locuteur met effectivement dans ses propos.

C'est également une méthode structurale car fondée sur le principe que les éléments du discours ne prennent sens que dans et par leurs relations entre eux : les éléments eux-mêmes ne permettent pas de découvrir la signification du discours. En effet, c'est de la mise en évidence des interrelations, de la manière dont les éléments sont structurellement reliés (associations, oppositions) que se dégage le sens du discours (et non de leur ordre ou fréquence d'apparition).

1.2 La relation de disjonction

Comme nous l'avons souligné, l'analyse structurale part du principe qu'aucun élément du discours ne trouve sa signification en lui-même : les éléments du discours ne prennent sens que dans et par leurs relations entre eux.

Au niveau le plus élémentaire, l'analyse structurale suppose la mise en relation de deux éléments du discours : il s'agit de **la relation de disjonction**.

La relation de disjonction doit respecter certaines règles.

- **Postulat de binarité**: la relation de disjonction unit deux et seulement deux termes du discours. A chaque terme d'une disjonction correspond donc un et un seul inverse.
- **Critère d'homogénéité**: les deux termes de la disjonction doivent se rapporter à une réalité commune appelée **axe sémantique**. En d'autres mots, les termes doivent être comparables : l'axe sémantique est un peu un dénominateur commun.
- **Critère d'exclusivité**: les deux termes mis en relation doivent être mutuellement exclusifs. On n'envisage aucune réalité qui possède à la fois les propriétés d'un terme et de son inverse.
- **Critère d'exhaustivité**: la réunion de ces deux termes doit recouvrir complètement l'axe sémantique. Les deux termes doivent être les seuls à présenter cette propriété commune qu'est l'axe sémantique.

La figure 1.2 donne une représentation générale de le relation de disjonction.

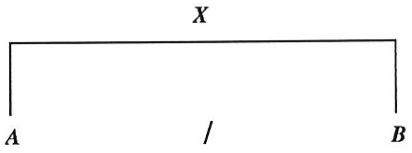


FIG. 1.2 – *Relation de disjonction entre les termes A et B. X est l'axe sémantique.*

On dit que A est l'inverse de B et réciproquement. Quand l'inverse d'un terme est la négation au sens grammatical (ou une forme équivalente), on parlera d'inverse **non-marqué**. Si ce n'est pas le cas, on parlera d'inverse **marqué** .

Un ou plusieurs termes d'une disjonction peuvent être connotés à l'aide d'un **indice de valorisation**. On affectera une valorisation opposée à l'inverse d'un terme valorisé.

Il arrive qu'un des termes de la disjonction (axe, inverse, valorisation) ne soit pas explicitement énoncé : on dit qu'il est **non-manifesté**. On formule alors une hypothèse quant à ce terme en explicitant le terme non manifesté. Nous soulignons cependant l'importance du fait que cette hypothèse doit respecter le point de vue du locuteur.

1.3 Les structures

Dans cette section, nous allons voir comment les disjonctions, éléments de base de l'analyse structurale, peuvent s'articuler entre elles pour former des **structures** plus complexes.

1.3.1 La relation d'implication et la structure parallèle

On dit qu'il y a une **relation d'implication réciproque** entre deux disjonctions quand les deux termes des deux disjonctions (A_1/B_1 et A_2/B_2) sont membre à membre unis par une relation de **double implication logique**. On a donc $A_1 \Leftrightarrow A_2$ et $B_1 \Leftrightarrow B_2$. On appelle **structure parallèle** une telle combinaison entre disjonctions.

La figure 1.3 représente une telle structure. Il faut remarquer que toutes les dispositions sont équivalentes.

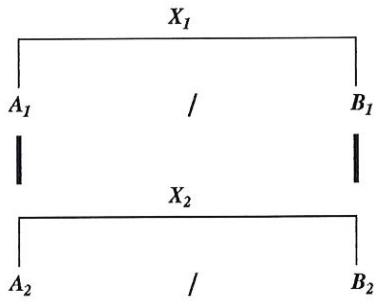


FIG. 1.3 – *La structure parallèle : X_1 (respectivement X_2) est l'axe sémantique relatif aux réalités A_1 et B_1 (respectivement A_2 et B_2). On a $A_1 \Leftrightarrow A_2$ et $B_1 \Leftrightarrow B_2$.*

Dans une structure parallèle, si un des termes est valorisé positivement,

alors tous les termes qu'il implique le sont de la même manière; l'ensemble des inverses est connoté de manière opposée.

1.3.2 La structure hiérarchisée

On peut rencontrer dans un discours des termes qui ont le double statut d'inverse d'une disjonction et d'axe d'une autre. La **structure hiérarchisée** rend compte de cette situation.

De manière générale, une structure hiérarchisée se présente comme à la figure 1.4. Nous avons donc une structure à plusieurs étages, chaque étage constituant un niveau hiérarchique.

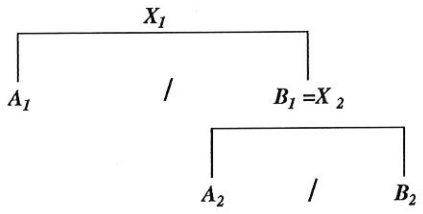


FIG. 1.4 – *La structure hiérarchisée : à un premier niveau, nous avons l'axe* X_1 *relatif aux réalités* A_1 *et* B_1*. Mais* B_1 *est également l'axe sémantique* $(B_1 = X_2)$ *d'une disjonction à un deuxième niveau de hiérarchie. Les deux réalités relatives à cet axe sont* A_2 *et* B_2*.*

Dans les structures hiérarchisées, chaque terme combine les connotations héritées des niveaux supérieurs et les connotations qui lui sont affectées à son propre niveau. En conséquence, plus un terme se trouve bas dans la structure, plus les valorisations peuvent devenir nombreuses. Sa valorisation peut alors se voir renforcée ou au contraire nuancée.

Evidemment, une structure hiérarchisée peut se composer de plusieurs disjonctions en cascade et combiner également des disjonctions en structure parallèle.

1.3.3 La structure croisée

La **structure croisée** permet de rendre compte des quatres possibilités de combinaison des termes de deux disjonctions.

Le croisement de ces disjonctions, appelées **disjonctions-mères** ou **axes-mères**, produit quatre quadrants correspondant aux quatre combinaisons possibles des termes prix deux à deux. Une représentation schématique de la structure croisée est donnée à la figure 1.5.

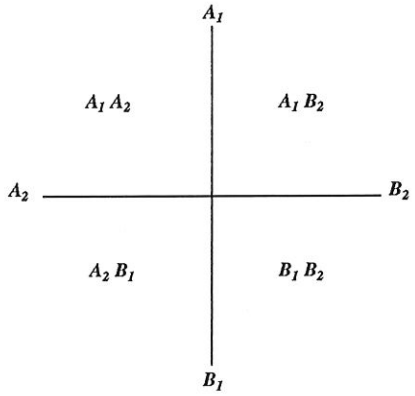


FIG. 1.5 – *La structure croisée : A_1 et B_1 (respectivement A_2 et B_2) sont les réalités-mères relatives à la première disjonction (respectivement à la seconde disjonction). A_1A_2, A_1B_2, A_2B_1 et B_1B_1 sont les réalités fécondées.*

Les termes des disjonctions dont les axes sont croisés sont appelés **réalités-mères**. Les réalités issues de la combinaison des axes sont appelées **réalités fécondées**. Aussi, chacune d'elles doit supposer les deux réalités-mères dont elle est issue.

Il convient alors de distinguer parmi les réalités fécondées :

- le nombre de réalités théoriquement possibles;
- le nombre de réalités prises en compte par le locuteur (qu'elles soient manifestées ou non). Par convention, on hachure la zone des réalités exclues;
- le nombre de réalités manifestées dans le matériau.

Quelquefois, les réalités fécondées peuvent se structurer entre elles, indépendamment de leur relation avec les axes-mères de la structure croisée. On peut les représenter par une seconde structure, complémentaire à la structure

croisée.

Les valorisations des axes de la structure (celles-ci suivant les règles des disjonctions classiques) se reportent sur les réalités fécondées. Il en résulte une réalité doublement positive, une réalité doublement négative et deux réalités ambivalentes. Si la réalité fécondée doublement positive est inaccessible, le choix doit se porter entre les deux réalités fécondées ambivalentes : c'est ce qu'on appelle le **dilemme**.

1.4 Exemples

Nous présentons pour terminer ce chapitre quelques exemples. Ceux-ci sont tous tirés de [10].

Exemple 1: ce premier exemple met en évidence une disjonction dans l'extrait suivant:

> *(Slogan des partis de la gauche lors de la campagne électorale pour les élections municipales à Marseille en 1983) La droite, c'est 20 ans d'immigration sauvage, la gauche, c'est l'immigration contrôlée.*

Cette disjonction est représentée à la figure 1.6

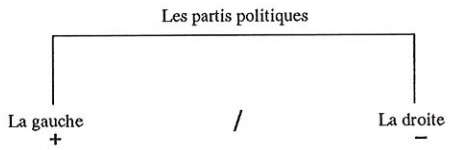


FIG. 1.6 – *Analyse de l'exemple 1.*

Exemple 2: considérons l'extrait

> *(interview de Patrick Bruel) Il y des films populaires qui sont moins dignes que d'autres, mais pour être un bon film, il faut aussi plaire au public. Pas seulement à quelques spécialistes.*

Nous pouvons relever la structure croisée représentée à la figure 1.7.

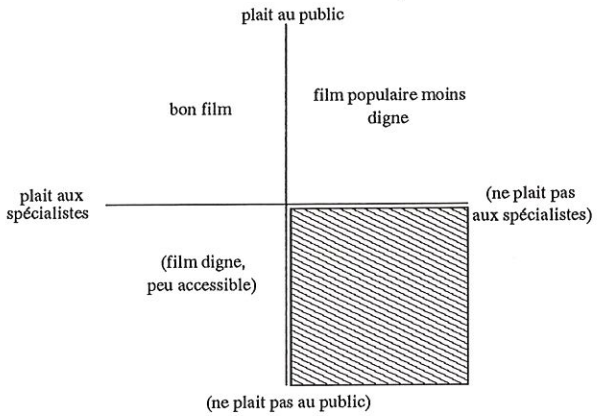


FIG. 1.7 – *Analyse de l'exemple 2.*

Exemple 3: une partie de l'analyse[3] de l'extrait suivant met en évidence une structure hiérarchique donnée à la figure 1.8.

> *(Extrait d'un discours de G. Deprez président du P.S.C. lors du congrès de Mouscron en 1983) S'il n'y pas d'avenir dans le démantèlement, il n'y en a pas non plus dans la nostalgie (...). Le fédéralisme, ce n'est pas seulement l'autonomie, c'est aussi l'union; ce n'est pas seulement le régional, c'est aussi le national. Nous sommes des fédéralistes d'union.*

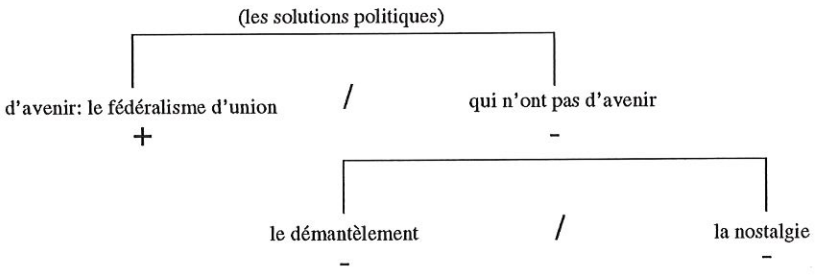


FIG. 1.8 – *Analyse de l'exemple 3.*

3. L'analyse complète de cet extrait est beaucoup plus compliquée. Le lecteur intéressé consultera [10] s'il désire l'analyse détaillée.

Disposant maintenant des notions et concepts de base de l'analyse structurale, nous pouvons nous "attaquer" à la modélisation 3D des structures présentées dans ce chapitre.

Chapitre 2

De l'analyse structurale à la visualisation en paysage

Nous présentons dans ce chapitre ce qui a motivé cette idée de visualisation des résultats de l'analyse strucurale en paysages. Nous développons également la modélisation graphique retenue. Ce chapitre constitue en fait une synthèse du cahier des charges du projet de création d'un logiciel d'analyse de contenu au départ des principes de l'analyse strucurale (projet FSR) rédigé par P. Alexandre [8].

2.1 Méthodologie du chercheur

Intéressons nous dans un premier temps à la manière de faire du chercheur lorsqu'il doit effectuer une analyse structurale dans le cadre de l'analyse du discours.

Dans un premier temps, le chercheur lit le texte à analyser et relève les différentes disjonctions.

Ensuite, le chercheur essaie différentes dispositions quant à la structure retenue en associant et en hiérarchisant ces disjonctions. Cette manière de faire lui permet de compléter le système de disjonctions.

La situation finale est en fait un arrangement des diverses disjonctions relevées et des structures successives que le chercheur leur a conféré. Le vé-

ritable défi en analyse structurale, est donc d'organiser les disjonctions qui pour leur part sont faciles à mettre en évidence.

Il nous faut insister sur le fait que puisque l'ensemble des disjonctions évolue en cours d'analyse, il vient un moment où il faut lister les disjonctions retenues, reprendre les associations existantes et structurer à nouveau l'ensemble. Cette démarche est récurrente et permet de la sorte d'enrichir le système de disjonctions. Cette étape constitue un véritable travail de fond et de réflexion pouvant durer plusieurs jours : c'est alors que mûrit l'analyse.

2.2 Vers une représentation en paysage

La démarche du chercheur se résume donc aux trois étapes suivantes :

- lecture attentive du texte,
- relevé des disjonctions,
- (première) structuration des disjonctions et enrichissement de la liste des disjonctions.

Lors de cette dernière étape, il existe essentiellement deux manières de procéder. Nous pouvons définir la structure d'un côté par une représentation en arbre ou d'un autre côté par une représentation en carte géographique.

2.2.1 La représentation en arbre

Cette manière de représenter les choses est classique et n'est pas propre à l'analyse structurale. Elle est connue dans la littérature sous l'appelation "*branches and nodes*". Le gros avantage de cette possibilité est son caractère répandu et relativement accessible. Ainsi, comme nous le verrons au chapitre 4, c'est sous cette forme que nous stockerons en mémoire dans l'ordinateur les données provenant de l'analyse structurale. Nous définissons d'ailleurs ci-dessous de façon informatique ce qu'est un arbre. Nous montrerons ensuite sur base de quelques exemples comment représenter la structure avec un arbre.

Formellement, un arbre peut être défini [1] de la manière suivante :

1. Un noeud unique, par lui-même est un arbre. Dans ce cas, il est aussi la racine de l'arbre.

2. Si n est un noeud et A_1, A_2, ..., A_k sont des arbres de racines respectives n_1, n_2, ..., n_k, on peut construire un nouvel arbre en associant comme parent unique aux noeuds n_1, n_2, ..., n_k le noeud n. Dans cette arbre, n est la racine et A_1, A_2, ..., A_k sont les sous-arbres de cette racine. Les noeuds n_1, n_2, ..., n_k sont appelés les fils de n.

Au niveau le plus élémentaire, une disjonction peut être représentée par un arbre dont la racine est le critère (l'axe de disjonction), et où les deux modalités constituent les deux fils de la racine. La figure 2.1 illustre ce point.

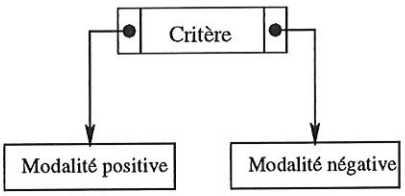


FIG. 2.1 – *L'axe de disjonction est la racine de l'arbre. Les deux modalités constituent les deux fils de l'arbre.*

Nous pouvons également représenter sous forme d'arbre une structure hiérarchique, comme nous le montre la figure 2.2.

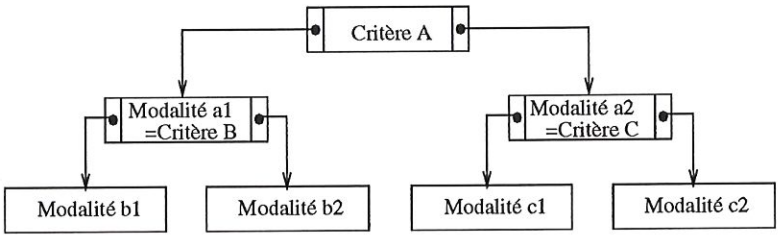


FIG. 2.2 – *Le premier axe de disjonction est la racine de l'arbre. Les deux modalités constituent les deux fils de l'arbre. La modalité de gauche et la modalité de droite sont à leur tour racines d'un arbre admettant deux fils.*

En ce qui concerne les structures croisées, plusieurs représentations sont possibles. Une possibilité est de représenter ce type de structure à l'aide d'une structure hiérarchique double comme illustré à la figure 2.3.

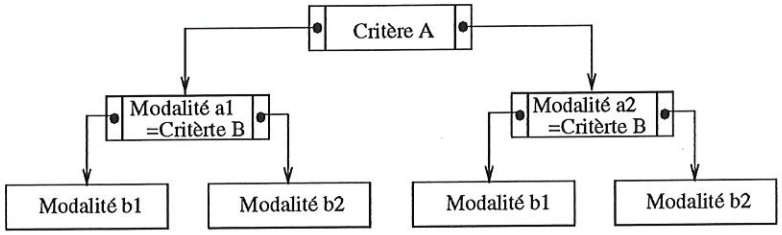


FIG. 2.3 – *Nous représentons une structure croisée à l'aide d'une structure hiérarchique double où chaque modalité de la première disjonction "pointe" vers le critère de la deuxième disjonction.*

La représentation[1] à laquelle nous avons pensé est la suivante. Nous savons que les réalités fécondées sont censées supposer les réalités-mères dont elles sont issues (voir la section 1.3.3). Dès lors, nous pouvons représenter une structure croisée à l'aide d'un arbre dont la forme est donnée à la figure 2.4.

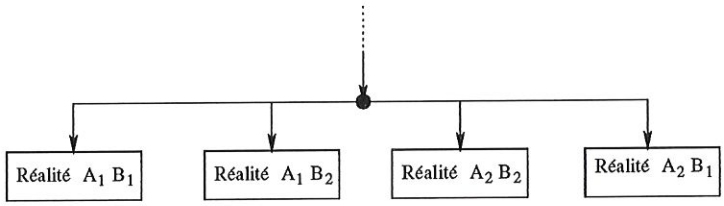


FIG. 2.4 – *Nous représentons une structure croisée par un noeud ayant quatre fils, chacun d'eux correspondant à une réalité fécondée. Il faut noter que ce noeud n'est ni un critère, ni une modalité : c'est en quelque sorte un noeud fictif dont le rôle est de nous informer que nous sommes face à une structure croisée.*

Nous avons préféré cette représentation à la première pour les raisons suivantes. Il nous semblait que le fait d'utiliser une structure hiérarchique double pour représenter une structure croisée pouvait donner lieu à des ambiguités au niveau de l'interprétation. En effet, il n'y a pas moyen de distinguer à

1. Cette représentation est propre à ce travail et n'est pas présentée dans le cahier des charges [8] contrairement aux autres représentations données dans ce chapitre. C'est d'ailleurs sous cette forme que seront stockées les structures croisées dans le programme.

priori une structure croisée d'une structure hiérarchique double[2] si nous utilisons ce type de représentation. Aussi, nous n'aimions pas l'idée que l'arbre représentant une structure croisée s'étale sur deux niveaux. Signalons tout de même que la représentation pour laquelle nous avons opté, a le désavantage que l'on "perd" les deux axes-mères, mais comme ceux-ci sont supposés par les réalités fécondées, nous avons estimé que ce n'était pas trop grave. Nous sommes donc amené à réaliser un compromis, chose fréquente lorsque nous cherchons à modéliser une situation.

La représentation en arbre présente cependant quelques inconvénients. Tout d'abord, elle isole la structure du contenu. Or, ce qui intéresse le chercheur, c'est le mode de structuration plutôt que la structure en soi : ce qu'il cherche à faire apparaître, c'est la manière dont s'organise le champ sémantique, c'est à dire le contenu dans ses systèmes de rapport. Ensuite, avec ce genre de représentation, les structures ont l'air de relier des îles perdues au milieu de nulle part[3], alors que la langue charie en tant que tels des systèmes d'évocation avec lesquels le locuteur compose. Aussi, il ressort l'idée que les structures s'ajoutent les unes aux autres, alors que le but est plutôt de montrer comment la structuration du champ sémantique se complexifie. Finalement, cette représentation accentue la vision binaire des choses. Une telle vision est réductrice du sens et donc inappropriée dans une démarche structuraliste visant à la production d'un sens.

La représentation à l'aide de cartes géographiques constitue un outil de structuration des disjonctions permettant de répondre en partie à ces limitations.

2.2.2 La représentation cartographique

Il s'agit ici d'un outil essentiellement graphique. Le chercheur agence dans l'espace selon un schéma propre à l'analyse structurale les différentes disjonctions relevées.

---

2. Nous pouvons très bien rencontrer dans certaines analyses des structures hiérarchiques doubles où les deux disjonctions du second niveaux sont identiques.

3. Extrait de [9] : *Structures look like nets linking isolated islands lost in the middle of nowhere.*

Ainsi, la figure 2.5 donne la représentation cartographique d'une disjonction.

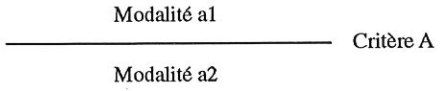


FIG. 2.5 – *Les deux modalités sont situées de part et d'autre d'une frontière représentant l'axe.*

Une structure croisée se met sous la forme donnée à la figure 2.6.

Nous signalons alors une convention que nous adopterons tout le reste de ce travail. Il fallait trouver un moyen d'identifier facilement les quatre "morceaux" d'une telle carte[4]. Nous procédons de la sorte: le "morceau" représentant la réalité fécondée doublement positive (resp. négative) sera appelé "morceau +" (resp. "morceau −"). Nous désignons finalement par "morceau +−" et "morceau −+" les deux quarts de la carte correspondant aux deux réalités ambivalentes.

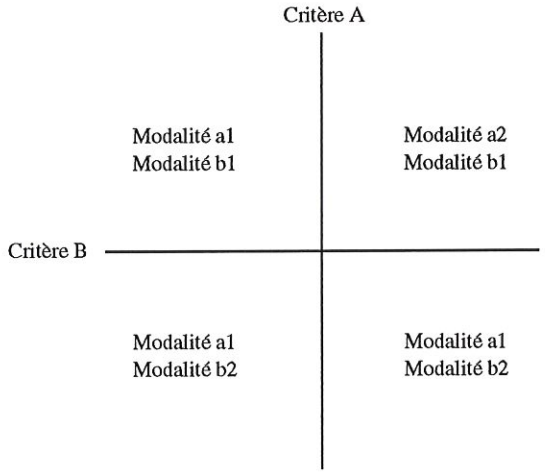


FIG. 2.6 – *Les deux axes partagent la carte en quatre (les réalités fécondées).*

4. Comme nous le verrons à la section 2.4, la représentation en paysage repose sur la représentation cartographique : il "suffit" d'ajouter des courbes de niveau (facteur hauteur).

La figure 2.7 montre le genre de représentation que l'on peut obtenir avec une structure plus compliquée.

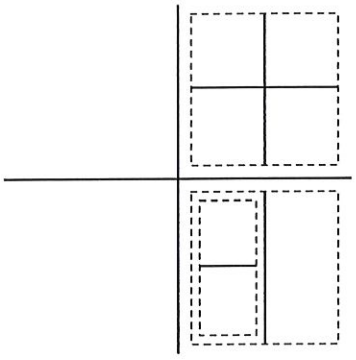


FIG. 2.7 – *Représentation cartographique de la structure suivante : nous avons tout d'abord une structure croisée. Dans celle-ci, une des réalités est une nouvelle structure croisée et une autre réalité, une disjonction. Une des modalités de cette disjonction se trouve être également une disjonction (structure hiérarchique). Afin de ne pas surcharger le dessin, nous n'avons pas indiqué les critères et les modalités.*

La représentation cartographique constitue une étape vers la représentation en paysage.

2.3 La représentation en paysage

Comme l'a souligné la section précédente, il n'est pas évident de s'accorder sur la représentation (ou sur la visualisation) de la structure. Ainsi, le chercheur se trouve souvent confronté à l'incompréhension quant aux idées que peuvent exprimer la représentation de son analyse. De plus, ces représentations ont des défauts susceptibles de fausser l'interprétation : trop forte idée de binarité, difficulté de percevoir les interrelations entre les mots, ...

Il fallait donc trouver une représentation des résultats plus parlante et aussi plus conviviale. Un long travail[5] de réflexion a finalement débouché sur l'idée de visualiser les résultats de l'analyse sous forme de paysages.

5. L'article [9] retrace "l'évolution" de l'enchainement d'idées qui a amené cette décision.

En effet, créer un paysage qui reflète les résultats de l'analyse rendrait certainement la représentation de la structure plus conviviale. De plus, la représentation en paysage permet de lever un grand nombre de problèmes liés aux représentations en arbre et en carte géographique.

Ainsi, dans une telle représentation, la forme du paysage (la hauteur des plateaux, les pentes, ...) nous semble significative en soi, avant même que nous ne prêtions attention au contenu linguistique du paysage (reflété par exemple à l'aide de mots écrits dans le paysage). On retrouve alors cette idée que le mode de structuration est plus important que la structure.

Le fait de séparer les deux modalités d'une disjonction par une montagne ou un ravin nous oblige à mettre ces deux modalités en relation : on ne peut concevoir une montagne qui n'a qu'un versant. De même, dans un paysage, cette impression de binarité apparaît beaucoup moins : il y a une continuité dans le paysage. Remarquons toutefois que l'analyse est toujours basée sur les disjonctions (et son postulat de binarité), mais dans un paysage, elles peuvent donner naissance à des formes variées.

L'utilisation du relief permet de rendre compte des valorisations. Dans notre civilisation, la hauteur est souvent assimilée à de bonnes et/ou importantes choses (les cieux constituant l'endroit le plus haut). Nous pouvons ainsi dans un paysage gérer la hauteur de chaque réalité en fonction de sa valorisation.

Une représentation en paysage permettra également de passer du point du vue du locuteur à celui de l'auditeur. Ainsi, le fait de se promener dans le paysage signifie que l'on découvre un monde. Nous pouvons alors tenter de comprendre ce monde au fur et à mesure de notre exploration. Nous pouvons également visualiser le paysage dans sa totalité (tel un oiseau survolant la campagne) et avoir un aperçu de la structure toute entière.

Enfin, l'ajout de couleurs, le degré de luminosité, les ombres, l'ajout d'objets (arbres, fleurs, ruisseaux, ..), le fait de pouvoir jouer sur les proportions des formes dans le paysage (altitude, largeur, longueur), ... permettront au chercheur de traduire sa sensibilité quant à l'analyse retenue.

2.4 Modélisation graphique

Comme nous l'avons laissé sous-entendre dans la section précédente, le paysage retenu sera composé de plateaux dont l'altitude relative sera fonction des valorisations sous-jacentes. Nous présentons dans cette section les situations que l'on peut rencontrer ainsi que la modélisation retenue pour ces situations. Nous verrons que cette modélisation permet de rencontrer à priori tous les types de structure générés par l'analyse structurale. Signalons déjà qu'une multitude de paramètres devront être ajustés afin de rendre le paysage le plus naturel possible.

2.4.1 Simple disjonction

La forme du paysage retenue pour modéliser une disjonction est donné à la figure 2.8. Sur cette figure, on constate que deux paramètres importants dans la création du paysage seront d'une part l'espace entre les modalités et d'autre part l'inclinaison des pentes (en plus bien évidemment de la hauteur ou profondeur relative des plateaux). Nous reviendrons plus en détail sur ces paramètres à la section 4.2.4 du chapitre 4.

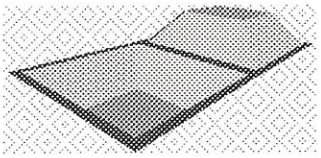


FIG. 2.8 – *Modélisation d'une disjonction.*

2.4.2 Simple croisée

Le paysage engendré par une structure croisée aura l'aspect général tel que nous le montre la figure 2.9.

2.4.3 Structures complexes

A titre d'exemples plus qu'à titre formel, nous montrons à la figure 2.10 (respectivement à la figure 2.11) les modélisations retenues si nous rencon-

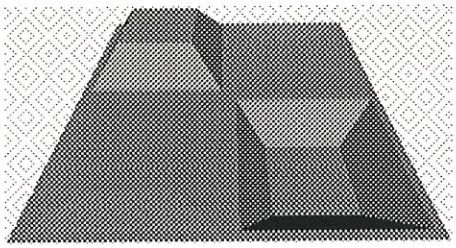


FIG. 2.9 – *Modélisation d'une structure croisée.*

trons une structure hiérarchique (respectivement des structures simples imbriquées).

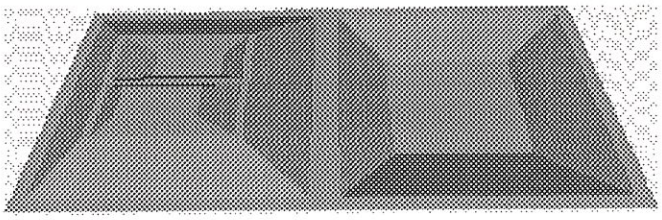


FIG. 2.10 – *Modélisation d'une structure hiérarchique.*

2.4.4 Cas particuliers

Nous n'avons envisagé pour le moment que la modélisation des cas "normaux". Qu'advient-il alors des cas particuliers sur lesquels peut aboutir une analyse?

Nous distinguons deux sortes de cas particuliers.

Tout d'abord, nous pouvons nous poser la question suivante: *comment synthétiser la modélisation quand il y a répétition d'une même sous-structure au sein d'une structure parent?* Nous renvoyons le lecteur au cahier des charges de l'application [8] s'il désire trouver des éléments de réponses à cette question[6].

6. Nous n'en disons pas plus ici car d'une part, la modélisation retenue pour la gestion

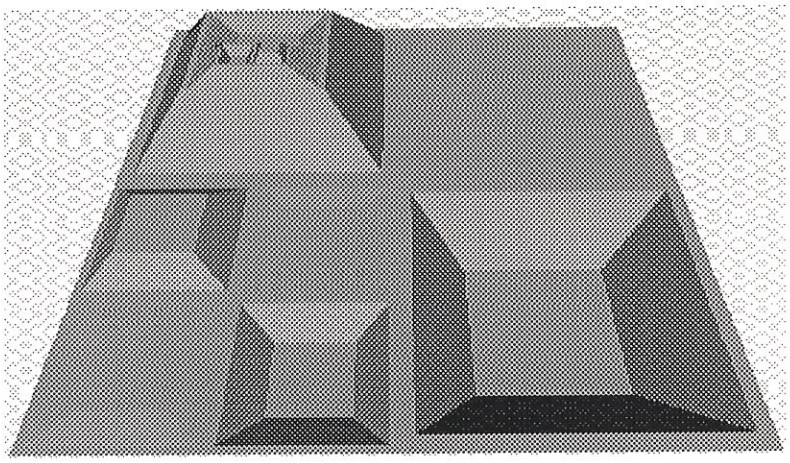


FIG. 2.11 – *Représentation en plateau de la structure décrite à la figure 2.7.*

Aussi, pour le moment, nous avons supposé qu'il n'y avait pas de réalité exclue ou de réalité inaccessible (**dilemme**) lorsqu'on a modélisé les structures croisées. Une autre question se pose alors : *comment représenter à l'aide du relief, d'un côté une structure croisée dont une des réalités est exclue et d'un autre côté une structure croisée où la réalité doublement positive est inaccessible ?* Le cahier des charges ne faisant pas état de ce dernier type de cas particuiers, nous ne nous y sommes pas attachés.

De toute façon, dans l'état actuel des choses, seuls des tests de visualisation pourront guider le chercheur afin qu'il élabore un modèle le plus proche de l'intuition et le mieux représentatif de l'analyse pour ces cas particuliers.

2.5 Adaptation de la modélisation graphique

Le problème de la génération des paysages se situe dans le fait que la représentation en plateaux telle que présentée à la section précédente est trop virtuelle. Ce à quoi on voudrait parvenir est la génération d'un paysage que l'on pourrait rencontrer dans la nature. Ainsi, nous pouvons dresser une liste de points dont il faudra tenir compte afin de rendre le paysage en plateau

de ces cas particuliers n'était pas définitive lors de la rédaction du cahier des charges et d'autre part nous n'avons pas géré ces cas dans le programme.

le plus naturel possible :

- lors de la génération du paysage, on devra pouvoir intégrer la notion d'arrondis. L'idéal serait de ne plus avoir d'arêtes, d'angles saillants ou de formes trop cubiques dans le paysage;
- introduire la notion d'aléatoire dans le paysage. En effet, cela permettra une construction moins géométrique, moins systématique de la structure. Aussi, cela permettra de générer un sol irrégulier, cahoteux;
- on devra pouvoir ajuster l'espace entre les structures, la surface des plateaux, la hauteur relative des plateaux;
- on devra pouvoir personnaliser le paysage : choix de la texture du sol, choix de l'ambiance de la scène, ajout d'objets dans le paysage (arbres, maisons, ruisseaux, ...). Ce décor rendra d'un côté le paysage plus réaliste et naturel, comme il permettra également au chercheur de traduire sa sensibilité quant à l'analyse retenue.

Le paysage présenté ci-dessous constitute la représentation en paysage de la structure hiérarchique mise en évidence dans l'exemple 3 à la section 1.4. Le reste de ce mémoire décrit les outils mis en oeuvre et les techniques utilisées pour atteindre ce genre de résultats.

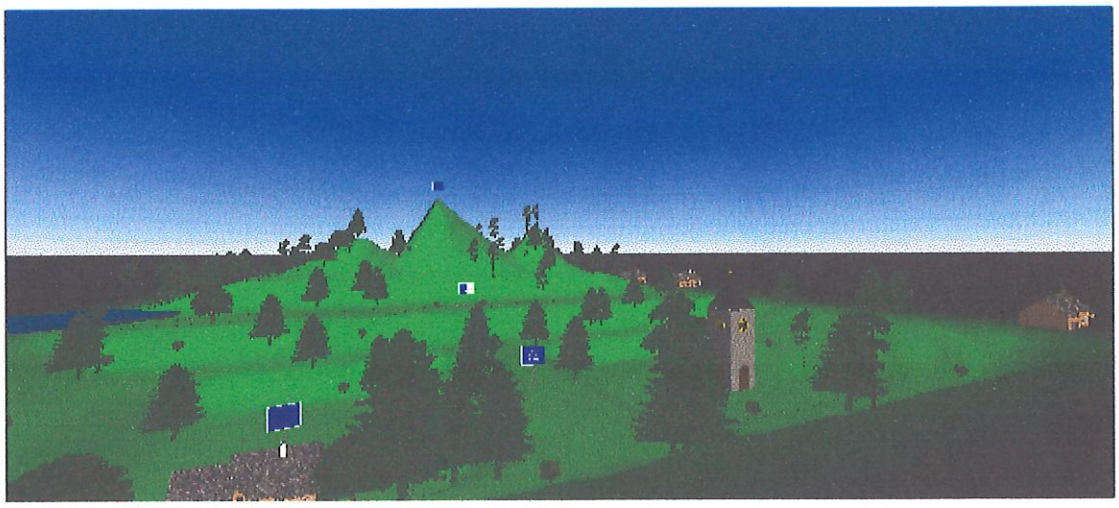


FIG. 2.12 – *Petit paysage généré par notre programme et représentant une structure hiérarchique.*

Chapitre 3

Rappels élémentaires de géométrie analytique

Toute structure peut être représentée par une carte (voir la section 2.2.2) à laquelle il faut ajouter des courbes de niveaux (les hauteurs déterminées par les valorisations (voir la section 2.4)). Le cadre de travail ne sera donc plus le plan mais l'espace 3D. Bien que celui-ci soit l'espace dans lequel nous évoluons, il n'est pas toujours simple de se représenter et de visualiser une scène en trois dimensions. Ce chapitre présente quelques notions élémentaires de géométrie bien utiles quand nous nous attaquons à la modélisation 3D.

Le but de ce chapitre est également de décharger le prochain chapitre des outils géométriques (et donc des développements mathématiques) utilisés dans le programme pour générer le paysage.

3.1 Le système de référence

Afin de situer exactement des points dans l'espace, nous avons besoin d'un **système de référence** ou de **coordonnées**. On utilise à cet effet un repère cartésien. Celui-ci est composé d'une origine O et de trois vecteurs normés linéairement indépendants notés e_1, e_2, e_3. On dit que ces vecteurs forment une base de l'espace (e_i est appelé $i$ème vecteur de la base canonique de $\mathbb{R}^3$). Nous appellerons l'axe X la droite passant par l'origine dans la direction e_1; l'axe Y la droite passant par l'origine dans la direction e_2; l'axe Z la droite passant par l'origine dans la direction e_3. Nous distinguons alors deux types

de repère :

- les repères gauchers : ce sont les repères tels que $e_1 \times e_2 = -e_3$,
- les repères droitiers : ce sont les repères tels que $e_1 \times e_2 = e_3$,

où l'opérateur $\times$ est l'opérateur produit vectoriel[1].

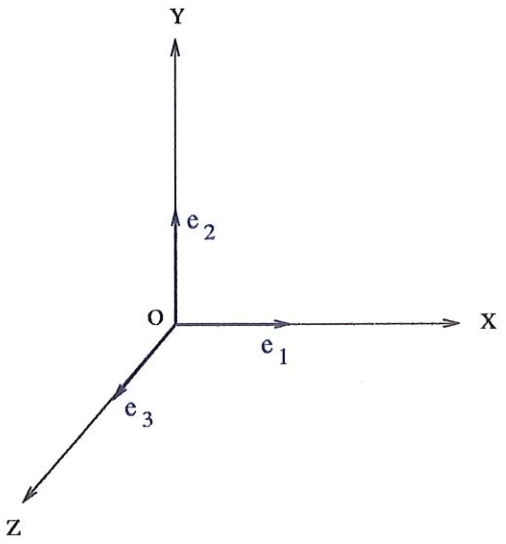


FIG. 3.1 – *Repère droitier*

Dans le cadre de ce mémoire, nous avons choisi de travailler avec un repère droitier, essentiellement parce que le language qui nous permettra de visualiser les paysages (*VRML*) utilise ce type de repère (se référer à l'annexe A).

1. Il existe un moyen facile pour déterminer si un repère est droitier ou gaucher. Le principe est le suivant. Placer le pouce de votre main droite dans la direction indiquée par l'axe X et l'index dans la direction indiquée par l'axe Y. Plier ensuite le majeur perpendiculairement à la paume de la main. Si la direction indiquée est celle de l'axe Z, le repère est droitier (voir la figure 3.1); si la direction indiquée est opposée à la direction de l'axe Z, le repère est gaucher. Cette petite règle est connue sous le nom de *règle de la main droite.*

3.2 Points et vecteurs

Un point (ou une position) de l'espace sera représenté par un triplet de réels. Ces trois réels sont les coordonnées du point dans le repère. Les points seront ensuite utilisés pour définir lignes, surfaces et polygones.

La définition de point telle que donnée précédemment permet de spécifier une position dans l'espace. Pour représenter une direction, nous utilisons des vecteurs. Ceux-ci sont également modélisés par un triplet de réels mais ils indiquent une direction dans l'espace plutôt qu'une position. En fait, ils indiquent la direction de la droite passant par l'origine et par le point associé au triplet.

3.3 Droites, plans et polygones

Maintenant que nous avons défini les outils de base que sont le repère, les points et les vecteurs, nous pouvons modéliser un grand nombre d'objets.

Nous commençons par modéliser la droite. Pour déterminer une droite, il suffit de connaitre un point de la droite et sa direction. Ainsi, si $P = (a,b,c)$ est ce point et $D = (A,B,C)$ cette direction, la droite a pour équation (dite paramétrique)

$$\begin{cases} x = a + uA \\ y = b + uB \\ z = c + uC \end{cases}$$

où u est un paramètre. P et D connus, nous pouvons dessiner la droite en déterminant les valeurs que prennent x, y et z pour différentes valeurs du paramètre u.

Exemple: un point P et une direction D nous permettent de retrouver la droite comme nous le montre la figure 3.2.

Nous pouvons également modéliser un grand nombre de courbes à l'aide d'équations dites paramétriques (le lieu des points définissant la courbe est déterminé en fonction de la valeur de un ou plusieurs paramètres). On trouve facilement dans la littérature (voir par exemple [4]) les équations paramétriques des courbes fréquemment rencontrées.

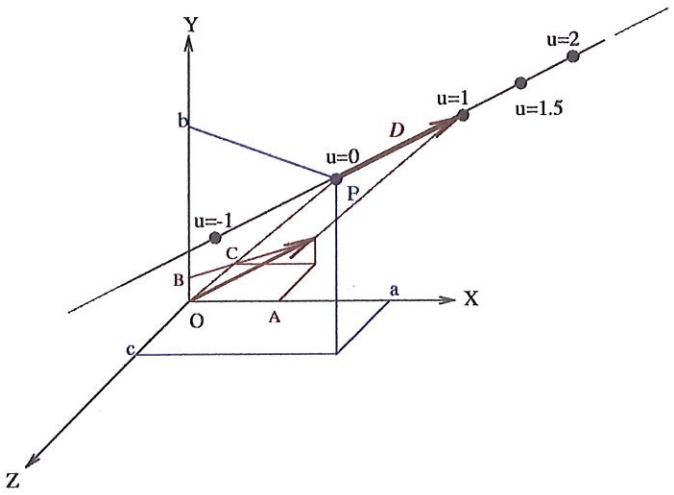


FIG. 3.2 – *Droite définie par un point P et une direction D.*

Un plan peut se définir également à l'aide d'une équation (dite cartésienne) :

$$\Pi \equiv Ax + By + Cz = d.$$

Le vecteur dont les composantes sont (A,B,C) est appelé vecteur normal au plan. Il s'agit d'un vecteur perpendicluaire au plan.

Nous pouvons de nouveau modéliser des surfaces plus compliquées qu'un plan à l'aide d'équations paramétriques. On se réferrera encore à la littérature pour de plus vastes explications.

Nous pouvons enfin caractériser un polygone en énumérant les sommets qui le composent.

3.4 Les transformations

Grâce à la définition du repère, des points et des vecteurs, nous sommes capables de déterminer la position et la représentation d'un objet tridimensionnel.

Nous pouvons alors effectuer des manipulations, appelées transformations, sur ces objets. Celles-ci incluent la translation, la rotation de centre, d'axe et d'angle donnés, la mise à échelle (équilibrée ou déséquilibrée), la projection sur une droite ou sur un plan et enfin la réflexion par rapport à un plan.

Toutes ces transformations s'appliquent aux points. Si nous désirons manipuler un objet, il suffit d'appliquer la manipulation à chacun des points qui le compose.

3.4.1 Translation

La translation déplace un point A de coordonnées (x,y,z) vers un point B de coordonnées (x',y',z') via un vecteur de composantes (tx,ty,tz) :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} tx \\ ty \\ tz \end{pmatrix}.$$

3.4.2 Rotation

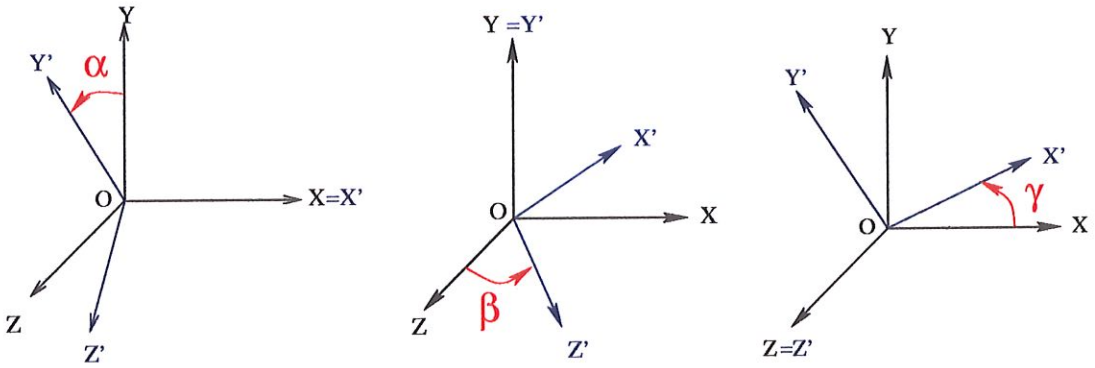
Dans un premier temps, nous allons définir les matrices de rotation lorsque le centre est l'origine et l'axe de rotation est soit l'axe X, soit l'axe Y, soit l'axe Z. Nous déterminerons ensuite la matrice de rotation quand le centre et l'axe sont quelconques.

Signalons que le sens de la rotation est donné par la règle dite du *tire-bouchon*[2].

Rotation d'un angle α autour de l'axe X.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

2. Cette règle dit que si nous attrappons l'axe de rotation avec la main droite, le pouce pointant dans la même direction que l'axe, alors le sens de rotation est le même que celui dans lequel sont enroulés les doigts autour de l'axe.



(a) Rotation d'un angle α autour de l'axe X.

(b) Rotation d'un angle β autour de l'axe Y.

(c) Rotation d'un angle γ autour de l'axe Z.

FIG. 3.3 – *Rotation de centre O autour des trois axes.*

Rotation d'un angle β autour de l'axe Y

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{3.1}$$

Rotation d'un angle γ autour de l'axe Z

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

Rotation de centre $C = (cx,cy,cz)$, d'axe $A = (ax,ay,az)$ et d'angle θ

Les trois rotations définies auparavant sont les plus courantes. Néanmoins, il peut arriver que nous voulions effectuer une rotation d'axe et de centre donné. Ce sera par exemple le cas si nous cherchons à modéliser un bras articulé, tel le pied d'une lampe de bureau (voir figure 3.4).

Nous allons donc déterminer une telle matrice de rotation. Nous nous appuierons sur la figure 3.5.

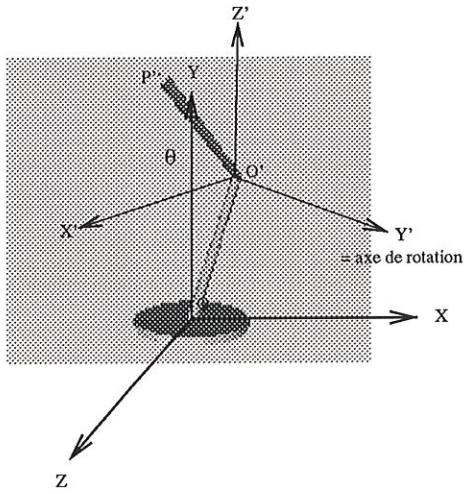


FIG. 3.4 – *Cette figure suppose qu'on a défini à un moment un cylindre centré à l'origine dont la génératrice est parallèle à l'axe X. Si nous désirons l'orienter de façon à représenter la deuxième moitié du bras articuler, on translate ce cylindre de sorte qu'une de ses extrémités soit située en O' et on effectue ensuite la rotation d'axe Y', de centre O' et d'angle θ.*

Sans perte de généralité, nous pouvons supposer que l'axe de rotation est normé. Nous pouvons également définir un nouveau repère orthonormé (O',e'_1,e'_2,e'_3) tel que la droite passant par O' et dans la direction e'_2 est notre axe de rotation.

La formule de changement de repère (passage du reprère (O,e_1,e_2,e_3) au repère (O',e'_1,e'_2,e'_3)) est :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathcal{M}[\begin{pmatrix} x \\ y \\ z \end{pmatrix} - b], \tag{3.2}$$

où

- b est le vecteur translation de l'origine (dans notre cas, le centre de rotation), c'est à dire $O' - O$ et
- $\mathcal{M} = \begin{pmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ z'_1 & z'_2 & z'_3 \end{pmatrix}$ où $\begin{pmatrix} x'_1 \\ y'_1 \\ z'_1 \end{pmatrix} = e'_1, \begin{pmatrix} x'_2 \\ y'_2 \\ z'_2 \end{pmatrix} = e'_2, \begin{pmatrix} x'_3 \\ y'_3 \\ z'_3 \end{pmatrix} = e'_3.$

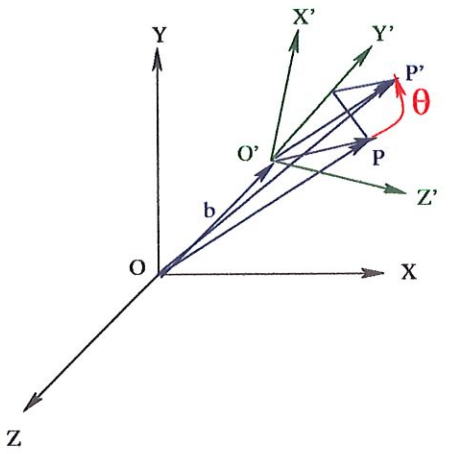


FIG. 3.5 – *Rotation de centre et d'axe quelconque*

Soit alors un point quelconque (x,y,z). Nous désirons appliquer une telle rotation à ce point. Nous commençons par exprimer les coordonnées de ce

point dans le repère (O',e'_1,e'_2,e'_3). Nous effectuons ensuite la rotation par rapport à O' autour de l'axe Y' et d'un angle θ. La matrice de rotation dans ce repère (notons la $\mathcal{R}$) est donnée par 3.1. Finalement, nous devons revenir dans le repère (O,e_1,e_2,e_3). Pour ce faire, on inverse la formule 3.2, ce qui donne

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = b + \mathcal{M}^T[\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}],$$

où nous avons tenu compte du fait que $\mathcal{M}$ est une matrice orthogonale (car matrice de changement de base orthonormale) et donc jouit de la propriété $\mathcal{M}^{-1} = \mathcal{M}^T$. Si nous reprenons toutes les transformations appliquées à (x,y,z), nous avons (en notant (rx,ry,rz) le point obtenu en fin de compte)

$$\begin{pmatrix} rx \\ ry \\ rz \end{pmatrix} = \mathcal{M}^T\mathcal{R}\mathcal{M}[\begin{pmatrix} x \\ y \\ z \end{pmatrix} - b] + b.$$

Il existe une manière plus simple de résoudre ce problème. Sans perdre de généralité, nous supposons que le centre de rotation est l'origine (si le centre n'est pas l'origine, on effectue une translation pour s'y ramener). Nous notons $\vec{a}$ l'axe de rotation et appelons D la droite passant par l'origine et de vecteur directeur $\vec{a}$.

Soit alors un point $\vec{p}$ quelconque. On désire calculer $R(\vec{p})$ où R désigne la rotation d'angle θ et d'axe $\vec{a}$.

On a $\vec{p} = \vec{p}_{//} + \vec{p}_{\perp}$ où $\vec{p}_{//}$ (respectivement $\vec{p_{\perp}}$) désigne la projection du vecteur $\vec{p}$ sur D (respectivement sur le plan perpendiculaire à D). Alors, comme $\vec{p}_{//}$ appartient à D, $R(\vec{p}_{//}) = \vec{p}_{//}$. Aussi, décomposons $R(\vec{p}_{\perp})$ sur les vecteurs $\vec{p}_{\perp}$ et $\vec{a} \times \vec{p}_{\perp}$. Nous obtenons : $R(\vec{p}_{\perp}) = \cos\theta\ \vec{p}_{\perp} + \sin\theta\ (\vec{a} \times \vec{p}_{\perp})$. Finalement, on a :

$$\begin{aligned} R(\vec{p}) &= R(\vec{p}_{//}) + R(\vec{p}_{\perp}) \\ &= \vec{p}_{//} + \cos\theta\ \vec{p}_{\perp} + \sin\theta\ (\vec{a} \times \vec{p}_{\perp}) \\ &= \cos\theta\ \vec{p} + (1 - \cos\theta)(\vec{p} \cdot \vec{a})\vec{a} + \sin\theta\ (\vec{a} \times \vec{p}) \end{aligned}$$

car $\vec{p}_{//} = (\vec{p} \cdot \vec{a})\vec{a}$ et $\vec{p}_{\perp} = \vec{p} - \vec{p}_{//}$. Nous avons donc une autre formule pour calculer une rotation d'axe quelconque. Dans le programme, nous avons utilisé cette formule.

3.4.3 La mise à échelle

La mise à échelle d'un point de coordonnées (x,y,z) correspond à la multiplication des ses trois composantes par un triplet de scalaires (a,b,c) :

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a \cdot x \\ b \cdot y \\ c \cdot z \end{pmatrix}.$$

Si $a = b = c$, la mise à échelle est dite équilibrée; elle est déséquilibrée sinon. Notons que la mise à échelle équilibrée préserve les formes et permettra de réaliser des homothéties avec les polygones.

3.4.4 Projection d'un point sur une droite

Soit D, une droite passant par le point (a,b,c) et de vecteur directeur (A,B,C); cette droite admet pour équation

$$\begin{cases} x = a + \alpha A \\ y = b + \alpha B \\ z = c + \alpha C \end{cases}$$

où α est un paramètre. Nous recherchons la projection d'un point $P = (s,t,u)$ sur la droite D. Comme le montre la figure 3.6, ce point est situé à l'intersection de la droite D et de la droite D' passant par P et dont le vecteur directeur est perpendiculaire à D.

L'équation de D' est donc la suivante

$$\begin{cases} x = s + \beta A' \\ y = t + \beta B' \\ z = u + \beta C' \end{cases}$$

où β est un paramètre et où A', B' et C' vérifient $AA' + BB' + CC' = 0$. En égalant les deux expressions pour x, y et z, on obtient

$$s + \beta A' = a + \alpha A \tag{3.3}$$

$$t + \beta B' = b + \alpha B \tag{3.4}$$
$$u + \beta C' = c + \alpha C \tag{3.5}$$

Multiplions alors (3.3) par A, (3.4) par B et (3.5) par C et sommons ces trois nouvelles équations. Comme $AA' + BB' + CC' = 0$, on obtient comme expresssion pour α (en supposant sans perte de généralité que $A^2+B^2+C^2 = 1$, i.e. que le vecteur directeur de D est normé)

$$\alpha = A(s-a) + B(t-b) + C(u-c). \tag{3.6}$$

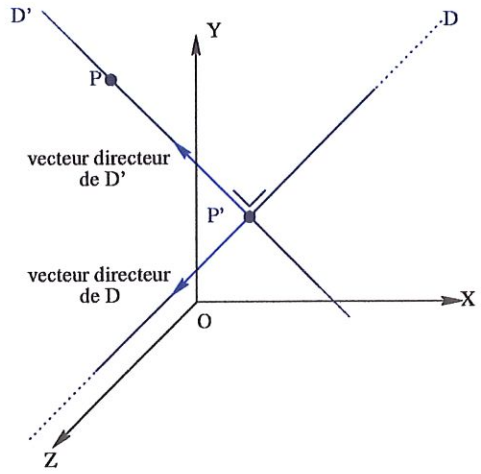


FIG. 3.6 – *Projection d'un point sur une droite.*

En injectant cette valeur de α dans l'équation de D, on obtient les coordonnées du point recherché

$$\begin{pmatrix} s' \\ t' \\ u' \end{pmatrix} = \begin{pmatrix} A^2 & AB & AC \\ AB & B^2 & BC \\ AC & BC & C^2 \end{pmatrix} \begin{pmatrix} s \\ t \\ u \end{pmatrix} + (Aa+Bb+Cc) \begin{pmatrix} A \\ B \\ C \end{pmatrix} + \begin{pmatrix} a \\ b \\ c \end{pmatrix}.$$

3.4.5 Projection d'un point sur un plan

Soit Π, un plan passant par $P = (a,b,c)$ et de vecteur normal $N = (A,B,C)$; Π admet alors pour équation

$$Ax + By + Cz = Aa + Bb + Cc.$$

Nous allons déterminer la formule qui nous permettra de calculer la projection (s',t',u') d'un point (s,t,u) sur le plan Π. Sur la figure 3.7, on s'aperçoit que (s',t',u') se situe à l'intersection du plan Π et de la droite D passant par (s,t,u) et de vecteur directeur N.

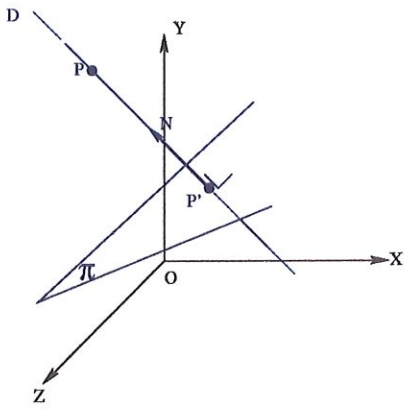


FIG. 3.7 – *Projection d'un point sur un plan.*

La droite D admet pour équation

$$D \equiv \begin{cases} x = s + \alpha A \\ y = t + \alpha B \\ z = u + \alpha C \end{cases}$$

où α est un paramètre. En injectant ces expressions pour x, y et z dans l'équation du plan, on obtient comme valeur de α (attention, ce n'est pas la même expression que(3.6))

$$\alpha = A(a - s) + B(b - t) + C(c - u) \tag{3.7}$$

où on a supposé sans perte de généralité que $A^2 + B^2 + C^2 = 1$, c'est à dire que le vecteur normal est unitaire. On obtient finalement la formule suivante :

$$\begin{pmatrix} s' \\ t' \\ u' \end{pmatrix} = \begin{pmatrix} 1 - A^2 & -AB & -AC \\ -AB & 1 - B^2 & -BC \\ -AC & -BC & 1 - C^2 \end{pmatrix} \begin{pmatrix} s \\ t \\ u \end{pmatrix} + (Aa + Bb + Cc) \begin{pmatrix} A \\ B \\ C \end{pmatrix}.$$

3.4.6 Symétrie par rapport à un plan

Il peut être parfois intéressant de trouver l'image d'un objet par une symétrie de plan donné. Pour ce faire, nous devons appliquer à tous les points qui définissent cet objet la transformation en question. Soit alors (s,t,u) un point; nous cherchons la formule qui nous permettra de déterminer (s',t',u'), point symétrique de (s,t,u) par rapport au plan Π d'équation

$$Ax + By + Cz = Aa + Bb + Cc.$$

Si nous reprenons ce qui a déjà été fait à la section précédente, on se rend compte que le point recherché est donné par

$$\begin{cases} s' = s + (2\alpha)A \\ t' = t + (2\alpha)B \\ u' = u + (2\alpha)C \end{cases}$$

où α est donné par 3.7. La figure 3.8 justifie cette assertion.

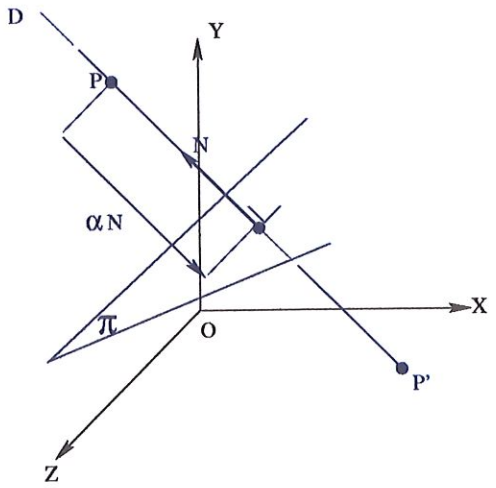


FIG. 3.8 – *Symétrie par rapport à un plan donné.*

Si nous désirons obtenir une formule qui ne dépend pas du point de départ, nous avons

$$\begin{pmatrix} s' \\ t' \\ u' \end{pmatrix} = \begin{pmatrix} 1-2A^2 & -2AB & -2AC \\ -2AB & 1-2B^2 & -2BC \\ -2AC & -2BC & 1-2C^2 \end{pmatrix} \begin{pmatrix} s \\ t \\ u \end{pmatrix} + 2(Aa+Bb+Cc) \begin{pmatrix} A \\ B \\ C \end{pmatrix}.$$

Tous les outils géométriques présentés dans ce chapitre seront d'une grande utilité dans le programme. Nous allons nous en rendre compte en lisant la suite.

Chapitre 4

Mise en oeuvre

Nous allons dans ce chapitre expliquer pas à pas le fonctionnement interne du programme que nous avons écrit afin de visualiser des structures venant de l'analyse structurale.

4.1 Langages utilisés

Nous avons utilisé comme langage de programmation *Visual C++*. Nous avons choisi de travailler en *C++* car c'est un langage rapide et puissant : *C++* fait en effet partie des langages dit "*orientés objets*". Comme nous le verrons plus loin, avec ce type de langage, on définit des objets ayant des propriétés. Nous pouvons ensuite manipuler ces objets à notre guise. En utilisant ce langage, nous avions aussi à notre disposition un programme écrit en *C++* traitant de la génération de vues de paysages 3D (voir [5]). Ce programme fut une référence très précieuse. Un avantage supplémentaire de *Visual C++*, est qu'il permet de réaliser aisément des interfaces utilisateurs multifenêtrées de type *Windows*[1].

En ce qui concerne la visualisation des paysages, nous avons utilisé le langage *VRML*. Ces quatres lettres signifient *Virtual Reality Modeling Langage* (Langage de Modélisation des Réalités Virtuelles). En effet, notre programme génère sur base des résultats de l'analyse structurale différents fichiers au format *VRML* (fichiers `.wrl`) contenant la description du paysage.

1. Il s'agit là en fait d'une des contraintes reprises dans le cahier des charges (voir [8]).

4.2 Génération du relief

Nous entendons par relief le paysage brut. En d'autres mots, il s'agit du paysage auquel le chercheur n'a pas encore apporté sa touche personnelle (ajout d'une texture de sol, ajout d'objets, ajout d'une ambiance, ...). Ce paysage est la traduction directe des résultats de l'analyse structurale en une représentation en plateau.

4.2.1 Modélisation des points et des faces

Pour stocker dans la mémoire d'un ordinateur des points ou des vecteurs, nous avons défini un objet *C++* appelé *CPoint3D*. En *C++* (comme dans tous les langages orientés objets d'ailleurs), un objet se compose d'attributs et de méthodes. Les attributs d'un objet sont en quelque sorte ses caractéristiques; les méthodes constituent l'ensemble des opérations que l'on peut effectuer sur cet objet. Ainsi, notre objet *CPoint3D* admet trois attributs (trois champs) correspondant aux composantes x, y et z d'un point (ou d'un vecteur). Nous avons également défini les méthodes suivantes :

- *Norme* : retourne la norme du vecteur (x,y,z), i.e. $(x^2+y^2+z^2)^{1/2}$;
- *Normer* : beaucoup de développements géométriques sont plus aisés quand on manipule des vecteurs normés. Nous avons implémenté une méthode normant le vecteur (x,y,z) : il suffit de diviser chacune des composantes par la norme du vecteur;
- *Translation* : permet de translater le point;
- *Rotation* : permet de faire subir à un point une rotation de centre, d'axe et d'angle donné. Comme les matrices de rotation dont le centre est l'origine et l'axe un des trois axes X, Y ou Z se déterminent plus aisément que les matrices de rotation de centre et d'axe quelconque, la méthode *Rotation* effectue des rotations différentes en fonction de ses arguments[2]. Si elle admet comme argument trois réels (notons les rx, ry et rz), alors le point effectuera d'abord une rotation d'un angle ry autour de l'axe Y, ensuite une rotation d'un angle rz autour de l'axe Z et finalement une rotation d'angle rx autour de l'axe X. Si elle admet comme argument deux objets de type *CPoint3D* et un réel,

2. Le *C++* autorise ce qu'on appelle la surcharge des fonctions. En d'autres mots, le *C++* offre la possibilité de définir des fonctions ayant le même nom mais des arguments différents.

alors le point effectuera une rotation de centre (respectivement d'axe) donné par le premier (respectivement second) objet de type *CPoint3D*. L'angle est donné par le réel. Nous utilisons les formules développées à la section 3.4.2 pour calculer le nouveau point;

- *Echelle*: permet une mise à échelle des composantes du vecteur (x,y,z). Nous multiplions en fait les trois composantes par trois scalaires;
- *ProjPlan*, *ProjDroite*: ces méthodes utilisent les formules développées au chapitre précédent pour trouver la projection d'un point sur un plan ou sur une droite;
- *Symetrie*: calcule le point symétrique par rapport à un plan d'un point donné.

Nous avons également défini des méthodes permettant de réaliser les opérations de base sur les vecteurs[3] :

- + (-) : opération d'addition (de soustraction) de deux vecteurs;
- $*$: produit scalaire de deux vecteurs;
- % : produit vectoriel de deux vecteurs.

Les faces que nous avons modélisées sont toutes rectangulaires. En effet, nous n'avons ressenti à aucun moment lors de l'implémentation le besoin de se définir d'autres types de face (triangulaires notamment). Ces faces rectangulaires nous serviront en fait essentiellement à créer la charpente[4] du paysage. L'implémentation des faces rectangulaires est toutefois suffisamment souple pour pouvoir être étendue à des faces plus générales si on en éprouve un jour le souhait.

Afin de représenter en mémoire ces faces rectangulaires, nous avons créé un autre objet *C++* appelé *CFace*. Cet objet possède les attributs suivants :

- *Pts*: il s'agit d'un pointeur sur un tableau de points (d'objets de type *CPoint3D*) définissant les (quatre) sommets de la face;

3. Le *C++* autorise également la surcharge des opérateurs. Le principe est le même que pour la surcharge des fonctions. La surcharge des opérateurs est toutefois limitée aux opérateurs correspondant à un symbole défini en *C++*. Ainsi par exemple, le symbole $*$ désigne à la base l'opérateur d'indirection. Par contre, le symbole $\times$ n'est pas défini en *C++* et donc nous avons choisi de désigner le produit vectoriel par % (division modulo).

4. On entend par charpente la forme générale du paysage. La représentation en plateau telle que présentée au chapitre 2 déterminera cette forme.

- *Info* : c'est un pointeur vers un champ d'information concernant la face. Ce champ contient les renseignements suivants :
  - le centre analytique de la face (*Centre*);
  - trois vecteurs (objets de type *CPoint3D*) formant un repère orthonormé associé à la face (*AxeX*, *AxeY*, *AxeZ*). Nous verrons plus loin comment est construit ce repère (voir figure 4.1);
  - la longueur et la largeur de la face[5] (*EchX*, *EchZ*);
  - un objet de type *CPoint3D* appelé *Angle* et dont les trois composantes x, y et z sont telles que si le repère de référence (O,e_1,e_2,e_3) effectue une rotation d'axe Y et d'angle y, une rotation d'axe Z et d'angle z et une rotation d'axe X et d'angle x, nous obtenons (à une translation près $P_1 - O$) le repère associé à la face. Ces trois réels nous fournissent en fait l'orientation de la face dans l'espace par rapport au plan XZ et par rapport à l'axe Y.

  Tous ces renseignements seront nécessaires lors de la génération du relief.

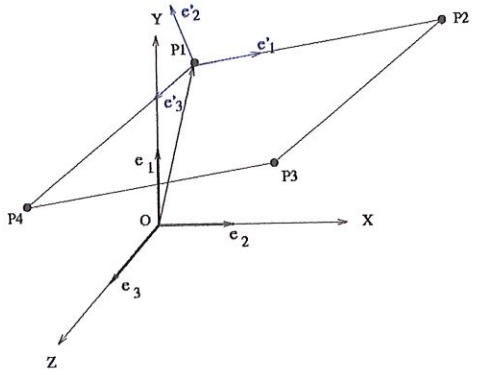


FIG. 4.1 – *Association d'un repère droitier* (e'_1,e'_2,e'_3) *à une face.*

Le type d'objet *CFace* admet alors les méthodes suivantes :

- *Init* : initialise les sommets de la face (i.e. *Pts*);

5. Par convention, la longueur (respectivement la largeur) est la dimension de la face dans le sens de l'axe X' (respectivement Y') du repère associé à la face. X' (resp. Y') est bien évidemment l'axe du repère parallèle au vecteur e'_1 (resp. e'_2).

- *CalcInfo* : calcule toutes les informations contenues dans le champ *Info*. Le calcul du centre analytique est aisé. Pour déterminer la valeur des autres champs, nous associons à la face un repère orthonormé droitier : nous choisissons comme vecteur e'_1 le vecteur allant du premier au deuxième sommet, comme vecteur e'_3 le vecteur allant du premier au dernier sommet. Le vecteur e'_2 est déterminé par $e'_1 \times e'_3$ (se référer à la figure 4.1). Nous trouvons ainsi les valeurs pour *AxeX*, *AxeY*, *AxeZ*. La longueur (respectivement la largeur) vaut alors la norme du côté parallèle au vecteur e'_1 (respectivement au vecteur e'_3). Pour déterminer les trois composantes de l'objet *Angle*, nous devons résoudre par rapport à α, β et γ le système d'équations suivant :

$$\begin{cases} \cos\gamma\cos\beta = x'_1 \\ \cos\alpha\sin\gamma\cos\beta + \sin\alpha\sin\beta = x'_2 \\ \sin\alpha\sin\gamma\cos\beta - \cos\alpha\sin\alpha = x'_3 \\ -\sin\gamma = y'_1 \\ \cos\alpha\cos\gamma = y'_2 \\ \sin\alpha\cos\gamma = y'_3 \\ \cos\gamma\sin\beta = z'_1 \\ \cos\alpha\sin\gamma\sin\beta - \sin\alpha\sin\beta = z'_2 \\ \sin\alpha\sin\gamma\sin\beta + \cos\alpha\cos\alpha = z'_3 \end{cases}$$

correspondant aux développements des équations

$$\begin{cases} \mathcal{R}_{X,\alpha}\mathcal{R}_{Z,\gamma}\mathcal{R}_{Y,\beta}e_1 = e'_1 \\ \mathcal{R}_{X,\alpha}\mathcal{R}_{Z,\gamma}\mathcal{R}_{Y,\beta}e_3 = e'_2 \\ \mathcal{R}_{X,\alpha}\mathcal{R}_{Z,\gamma}\mathcal{R}_{Y,\beta}e_3 = e'_3 \end{cases}$$

où $\mathcal{R}_{.,.}$ désigne une matrice de rotation de centre (0,0,0), autour d'un des trois axes et d'un angle donné. La résolution de ce système se fait au cas pas cas et est plus facile qu'il n'y paraît.

Nous retrouvons également pour les faces des méthodes permettant la translation, les rotations et la mise à échelle. Le principe de ces méthodes est simple : si nous désirons effectuer une des transformations citées ci-dessus, il suffit d'appliquer cette transformation à tous les sommets de la face. Il ne faut toutefois pas oublier de recalculer ensuite les différents champs d'information.

4.2.2 Gestion hiérarchique de la structure et du relief

Nous allons dans cette section présenter comment nous stockons en mémoire dans l'ordinateur d'une part les données provenant de l'analyse structurale et d'autre part le paysage généré. Nous verrons qu'il y a un parallélisme entre les deux, ce qui semble assez intuitif puisque nous cherchons à représenter une structure sous forme de paysage.

Gestion de la structure et format des fichiers de données

Nous avons choisi d'utiliser la représentation en arbre pour stocker la structure en mémoire. Néanmoins, le type d'arbre retenu diffère quelque peu de celui présenté à la section 2.2.1. Ainsi, nous avons créé un objet récursif *CArbre*. Ce type d'objet contient :

- *Niveau* : entier renseignant le niveau dans la structure auquel nous nous trouvons;
- *Valorisation* : vaut $+1$, 0 ou -1 selon que la valorisation de la structure actuelle est positive, non renseignée ou négative;
- *NbStr* : indique le nombre de sous-structures qu'admet la structure actuelle;
- Trois champs de texte (*MotCle*, *Expl*, *Lien*) qui vont permettre de maintenir une connexion avec le texte initial analysé. Ainsi, *MotCle* contiendra soit le nom d'un critère, soit d'une modalité. Nous avions émis le souhait de pouvoir afficher du texte dans le paysage final : *MotCle* servira à réaliser cet objectif. Le champ *Expl* (abréviation de Explication) est en fait complémentaire au champ *MotCle*. Il servira essentiellement à fournir quelques mots d'explication quant au texte contenu dans *MotCle* (reportez-vous au chapitre 5 pour de plus amples informations concernant ce champ). Enfin, le champ *Lien* permettra de rendre le paysage interactif. En effet *Lien* pourra contenir l'adresse de n'importe quel type de document[6] en rapport avec un critère ou une modalité et vers lequel nous désirons avoir un lien à partir du paysage. Dans le paysage final, le texte contenu dans *MotCle* apparaîtra dans des panneaux. En cliquant sur ces panneaux, l'utilisateur sera renvoyé vers le document dont *Lien* précise l'adresse;

6. Cela peut être un document texte contenant par exemple le texte de départ; cela peut être l'adresse d'un site Internet; cela peut être l'adresse d'un autre paysage, ...

- *FPos*, *FNeg*, *FPsNg*, *FNgPs*: ce sont quatre pointeurs, chacun vers un nouvel objet de type *CArbre*. *FPos* "pointe" vers l'éventuelle structure positive (dans le cas d'une simple disjonction ou dans le cas d'une structure croisée); *FNeg* "pointe" vers l'éventuelle structure négative (dans le cas d'une simple disjonction ou dans le cas d'une structure croisée); *FPsNg* (respectivement *FNgPs*) "pointe" vers l'éventuelle structure ambivalente $(+-)$[7] (respectivement $(-+)$) (dans le cas d'une structure croisée).

Schématiquement, la structure est stockée en mémoire sous la forme illustée à la figure 4.2.

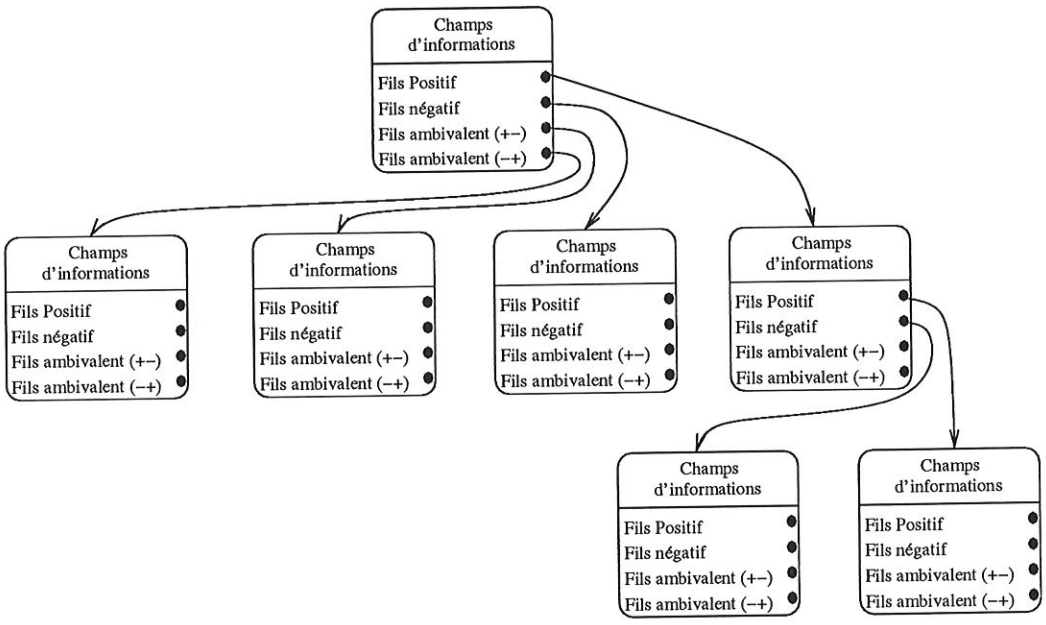


FIG. 4.2 – *Gestion hiérarchique de la structure (CPaysage et CArbre).*

Nous allons maintenant présenter le format des fichiers de données. En effet, en attendant l'implémentation d'une interface d'encodage de la structure telle que spécifiée dans le cahier des charges [8], nous avons élaboré un petit langage[8] permettant d'encoder assez facilement la description de la

7. D'après notre convention fixée à la section 2.2.2.

8. Ce langage est une adaptation du petit langage écrit par J. et S. Grad [5] dans le cadre de leur mémoire et destiné à simplifier la description de scènes 3D.

structure. Celui-ci est composé de 17 mots-clés[9] :

- `#NB_TOT` nb : ce mot-clé doit se trouver obligatoirement à la première ligne du fichier et indique le nombre total de structures;
- `#COM_D` : marque le début d'un commentaire;
- `#COM_F` : marque la fin d'un commentaire;
- `#NUM` num : numéro servant à identifier la structure de façon unique[10];
- `#VAL` Valorisation : valorisation de la structure;
- `#NIV` Niveau : niveau dans la structure globale auquel se trouve la structure actuelle (que nous sommes est train de décrire);
- `#NB_ST` NbStr : nombre de structures vers lesquelles pointe la structure actuelle;
- `#STP` p : indique le numéro de l'éventuelle structure valorisée positivement vers laquelle pointe la structure actuelle;
- `#STN` p : indique le numéro de l'éventuelle structure valorisée négativement vers laquelle pointe la structure actuelle;
- `#STPN` p : indique le numéro de l'éventuelle structure ambivalente $(+-)$ vers laquelle pointe la structure actuelle;
- `#STNP` p : indique le numéro de l'éventuelle structure ambivalente $(-+)$ vers laquelle pointe la structure actuelle;
- `#MOT_CLE` : marque le début du texte que contiendra le champ *MotCle*;
- `#EXPL` : marque le début du texte que contiendra le champ *Expl*;
- `#LIEN` : marque le début du texte que contiendra le champ *Lien*;
- `#FIN` : marque la fin du texte qui suit un mot des mots-clés `#MOTCLE`, `#EXPL`, `#LIEN`;
- `#SUIV` : mot-clé figurant en fin de description d'une structure si celle-ci n'était pas la dernière.

Exemples[11] :

Reprenons les exemples 2 et 3 du chapitre 1. L'exemple 2 consitait en l'analyse du matériau suivant

> *(interview de Patrick Bruel) Il y des films populaires qui sont moins dignes que d'autres, mais pour être un bon film, il faut aussi plaire au public. Pas seulement à quelques spécialistes.*

9. Ces mots-clés doivent être écrits en majuscules.
10. Attention: la numérotation commence à 0 et doit être continue.
11. Tous ces exemples sont tirés de [10].

```
#NUM 2
#NIV 1 #COM_D Nous sommes au niveau un #COM_F
#VAL -1 #COM_D Valorisation négative #COM_F
#MOT_CLE Les mauvais films #FIN
#EXPL Ne plait ni au public ni aux spécialistes (réalités exclues) #FIN
#LIEN analyse.txt #FIN
#SUIV #COM_D Fin de la description de la réalité fécondée négative #COM_F

#NUM 3
#NIV 1 #COM_D Nous sommes au niveau zéro #COM_F
#VAL 0 #COM_D Valorisation nulle car ambivalente #COM_F
#MOT_CLE Les films populaires moins dignes #FIN
#EXPL Plait au public mais pas aux spécialistes #FIN
#LIEN film.jpeg #FIN #COM_D film.jpeg est une image contenant une affiche
d'un tel type de film (par exemple) #COM_F
#SUIV
#COM_D Fin de la description de la réalité fécondée ambivalente (+-)#COM_F

#NUM 4
#NIV 1 #COM_D Nous sommes au niveau zéro #COM_F
#VAL 0 #COM_D Valorisation nulle car ambivalente #COM_F
#MOT_CLE Les films dignes mais peu accessibles #FIN
#EXPL Ne plait pas au public mais plait aux spécialistes #FIN
#LIEN analyse.txt #FIN
```

L'exemple 3 faisait intervenir une structure hiérarchique (figure 4.4) dans l'extrait suivant[12]:

> *(Extrait d'un discours de G. Deprez président du P.S.C. lors du congrès de Mouscron en 1983) S'il n'y pas d'avenir dans le démantèlement, il n'y en a pas non plus dans la nostalgie (...). Le fédéralisme, ce n'est pas seulement l'autonomie, c'est aussi l'union; ce n'est pas seulement le régional, c'est aussi le national. Nous sommes des fédéralistes d'union.*

Un fichier de données possible encodant cette analyse pourrait contenir les lignes suivantes :

```
#NB_TOT 5
```

12. Nous rappelons que l'analyse de matériau est plus compliquée que ce que nous allons en dire, mais notre but est avant tout ici de montrer comment concevoir correctement un fichier de données.

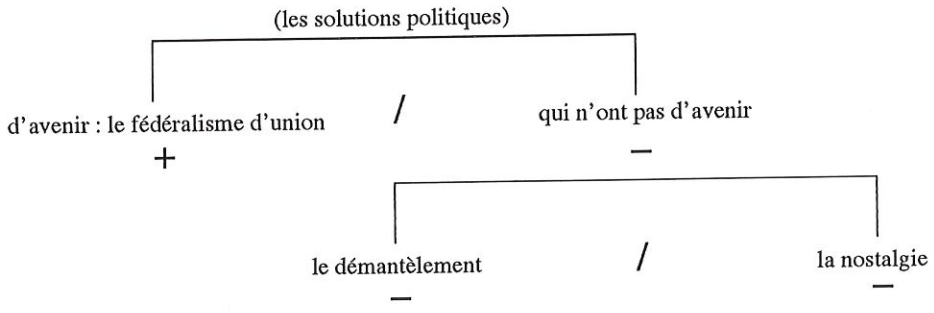


FIG. 4.4 – *Analyse du matériau 2.*

```
#COM_D Nous avons une structure hiérarchique #COM_F

#NUM 0
#NIV 0 #COM_D Nous sommes au niveau zéro #COM_F
#VAL 0 #COM_D Pas de valorisation #COM_F
#NB_ST 2 #STP 1 #STN 2 #COM_D Il y a deux sous-structures #COM_F
#MOT_CLE Les solutions politiques #FIN
#COM_D On n'est pas obligé de mettre un champ #EXPL #COM_F
#LIEN document.txt #FIN
#SUIV

#NUM 1
#NIV 1 #COM_D Nous sommes au niveau un #COM_F
#VAL 1 #COM_D Valorisation positive #COM_F
#MOT_CLE Les solutions politiques d'avenir : le fédéralisme d'union #FIN
#LIEN analyse.txt #FIN #COM_D analyse.txt contient l'analyse #COM_F
#SUIV

#NUM 2
#NIV 1 #COM_D Nous sommes au niveau un #COM_F
#VAL -1 #COM_D Pas de valorisation #COM_F
#NB_ST 2 #STP 2 #STN 4
#COM_D La modalité négative est critère d'une seconde disjonction #COM_F
#MOT_CLE Les solutions sans avenir #FIN
#LIEN analyse.txt #FIN
#SUIV

#NUM 3
#NIV 2 #COM_D Nous sommes au niveau deux #COM_F
#VAL -1
#MOT_CLE Le démantèlement #FIN
```

```
#EXPL Les deux modalités sont négatives #FIN
#LIEN analyse.txt #FIN
#SUIV

#NUM 4
#NIV 2 #COM_D Nous sommes au niveau deux #COM_F
#VAL -1
#MOT_CLE La nostalgie #FIN
#EXPL Les deux modalités sont négatives #FIN
#LIEN analyse.txt #FIN
```

Gestion hiérarchique du relief

Au fur et à mesure que nous parcourons un objet de type *CArbre*, nous construisons le relief, celui-ci constituant la représentation en paysage de la structure déjà explorée. Nous avons défini un type *CPaysage* pour stocker dans la mémoire de l'ordinateur le relief. Celui-ci a la même structure globale en arbre que le type *CArbre*.

Nous comprendrons mieux le rôle des divers attributs d'un objet de cette classe après avoir jeté un rapide coup d'oeil sur l'idée retenue pour générer le relief. Le principe est le suivant. Nous positionnons dans un premier temps des faces dans l'espace en fonction du niveau, de la valorisation et du type de la structure considérée (ces faces constituent en fait les sommets des plateaux). Dans un second temps, nous calculons des courbes de niveau entre ces différentes faces de sorte que le relief soit le plus naturel possible.

Nous pouvons maintenant décrire les divers champs d'un objet de type *CPaysage*:

- *Territoire*: pointeur sur un objet de type *CFace*. *Territoire* délimite l'espace dans lequel prendra forme le paysage représentatif de la structure actuelle;
- *SousTerritoire*: pointeur sur un tableau d'objets de type *CFace*. Chacune de ces faces détermine les bases des divers reliefs. La somme de ces faces ne vaut pas tout à fait la surface *Territoire*: nous avons ainsi un espace entre les structures;
- *SousPaysage*: pointeur sur un tableau d'objets de type *CFace*. Ces faces constituent les sommets des plateaux. Nous calculons les courbes de niveau entre les faces contenues dans le tableau vers lequel pointe

SousTerritoire et les faces contenues dans le tableau vers lequel pointe *SousPaysage*. Notons que ces faces constitueront les territoires pour les structures du niveau suivant;

- *CourbeNiveau*: pointeur sur un objet de type *CListe*. Nous avons en effet défini un type *Cliste*: il s'agit de l'implémentation du type de donnée appelé liste chaînée[13] , adapté à nos besoins, à savoir le stockage dynamique d'objets de type *CPoint3D*. Les points contenus dans cette liste définissent les courbes de niveau entre les faces décrites précédemment;
- *StPos*, *StNeg*, *StPsNg*, *StNgPs*: pointeurs vers des objets de type *CPaysage* (récursif). *StPos* (respectivement *StNeg*) pointe vers l'objet *CPaysage* représentant la structure positive (respectivement négative) (lors d'une disjonction ou d'un structure croisée); *StPsNg* (*StNgPs*) pointe vers l'objet de type *CPaysage* représentant la structure ambivalente $(+-)$ (respectivement $(-+)$) (lors d'une structure croisée).

Nous avons alors les méthodes suivantes[14] :

- *CPaysage*: constructeur[15] de la classe. Nous initialisons tous les champs de l'objet *CPaysage* à NULL[16];
- $\sim$*CPaysage*: destructeur de la classe[17];
- *CalcSousTerr*: détermine en fonction de divers paramètres, les faces (dimensions, position et orientation) contenues dans le tableau vers le quel pointe *SousTerritoire*;
- *CalcSousPays*: détermine en fonction de divers paramètres les faces (dimensions, position et orientation) contenues dans le tableau vers le

13. Une liste chaînée peut être défini [1] comme une liste de cellules. Chaque cellule comprend deux champs, l'un contenant un élément de la liste de départ et l'autre un pointeur vers la prochaine cellule de la liste chaînée.

14. Nous détaillons ces méthodes à la section suivante.

15. Dans la plupart des langages orientés objets , on a des constructeurs de classe. On appelle constructeur une fonction d'initialisation portant le même nom que la classe (il faut savoir que la déclaration d'un objet d'une certaine classe n'initialise pas celui-ci).

16. Nous avons vu que tous les champs de *CPaysage* étaient des pointeurs vers d'autres objets. NULL est la valeur d'un pointeur qui ne pointe sur rien du tout.

17. Lorsqu'un objet d'une classe est créé, le programme prend le contrôle de celui-ci aussi longtemps que celui-ci existe. Lorsque sa durée de vie se termine, le programme appelle automatiquement une fonction qui détruit celui-ci. Une telle fonction est appelée destructeur. La définition d'un destructeur n'est pas obligatoire (*C++* en génère un par défaut s'il n'y en a pas) mais s'avère utile dans le cas d'objets dynamiques (tel l'objet *CPaysage)*.

quel pointe *SousPaysage*;

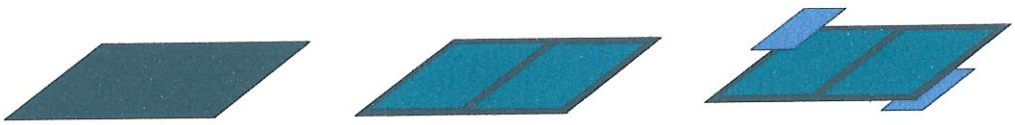
- *CalcCdNiv*: calcule les courbes de niveau entre les faces contenues dans le tableau vers lequel pointe *SousTerritoire* et celles contenues dans le tableau vers lequel pointe *SousPaysage*;
- *CalcPlan*, *CalcDome*, *CalcDent*: méthodes calculant les sommets des plateaux. Ces sommets peuvent être plats, en dôme ou dentelés;
- *CalcRelief*: parcourt récursivement l'objet *CPaysage* et crée une liste contenant tous les points qui vont générer le relief.

4.2.3 Génération du relief

Maintenant que nous avons introduit tous les objets qui nous paraissaient utiles, nous pouvons les manipuler dans le but final d'obtenir un paysage traduisant l'analyse le mieux possible.

Les méthodes CalcSousTerr et CalcSousPays

Pour chaque niveau dans l'arbre contenant l'analyse (i.e. dans un objet de type *CArbre*), nous traduisons la structure rencontrée sous la forme de plateaux (i.e. nous construisons un objet de type *CPaysage*). Ainsi, la première étape consiste en l'agencement dans l'espace de faces en respectant la modélisation en plateaux retenues.



(a) Territoire (b) Sous-territoires (c) Sous-paysages

FIG. 4.5 – *Les trois étapes de la génération de la représentation en plateau.*

Tout d'abord, nous plaçons une face (notons la T) qui délimite l'étendue du paysage (figure 4.5(a)).

Ensuite, en fonction du type de la structure, nous définissons un certain nombre de faces. Par exemple, si nous avons à faire à une simple disjonction,

nous définissons deux faces (notons-les T_+ et T_-) (figure 4.5(b)); si nous sommes en présence d'une structure croisée, nous définissons quatre faces (notons-les T_+, T_-, T_{+-} et T_{-+}). Ces faces déterminent la base des plateaux et permettent l'introduction d'un "no man's land". Nous pouvons agir sur les dimensions, l'orientation et la position de ces faces, ce qui permet déjà à ce niveau d'éviter une construction trop systématique et géométrique des plateaux. Dans le cadre de ce travail, nous avons choisi d'introduire simplement une composante aléatoire sur la dimension de ces faces. Nous avons cependant mis en place tous les outils nécessaires afin que le programme puisse être modifié afin que le chercheur puisse lui-même déterminer les dimensions, l'orientation et la position de ces faces.

Remarques: il y a tout de même des contraintes (assez logiques) sur la définition de ces faces. En effet, ces faces doivent être disjointes et contenues dans la face délimitant le paysage. Cela laisse cependant un degré de liberté assez important.

Une fois ces faces déterminées, nous calculons ensuite les faces qui vont constituer les sommets des plateaux (figure 4.5(c)). La position de ces nouvelles faces est déterminée dans un premier temps en fonction des valorisations (action au niveau de la hauteur) et en fonction des faces constituant les bases des plateaux définies auparavant (action sur la dimension). Dans un second temps, le chercheur peut lui aussi agir sur ces faces en modifiant certains paramètres. Il peut ainsi agir sur la hauteur de la face (et donc sur la hauteur relative des plateaux) ainsi que sur ces dimensions. Il peut de la sorte agir à sa guise sur les pentes des plateaux.

Remarques: les dimensions de ces faces "sommets" doivent être plus petites que les dimensions des faces "bases" correspondantes. Aussi, la projection de chacune des faces "sommets" dans le plan de sa face "base" correspondante doit être contenue dans celle-ci.

Nous obtenons en fin de compte une représentation en plateau qui est encore très géométrique. La détermination des courbes de niveau va résoudre ce problème.

La méthode CalcCdNiv

Nous disposons donc à ce stade de faces situées dans l'espace. Nous désirons déterminer des courbes joignant ces faces de sortes que le paysage final ait perdu cet aspect trop cubique.

Considérons alors les différentes courbes joignant deux points dans l'espace représentées à la figure 4.6.

Ce petit dessin nous montre deux points dans l'espace et quelques courbes joignant ces deux points. Nous constatons que la courbe qui semble la plus naturelle pour joindre ces deux points de sorte que les angles soient les moins saillants possibles est une courbe qui admet une dérivée nulle aux deux points et qui change de convexité une seule fois entre les deux points (la courbe (d)).

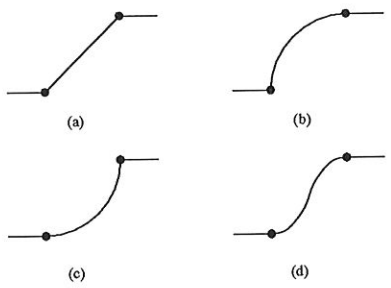


FIG. 4.6 – *Différentes courbes joignant deux points.*

Regardons alors le comportement de la fonction $\sin x$ dans le plan (figure 4.7).Comme nous le constatons la fonction $\sin x$ admet une dérivée nulle en $x = -\pi/2$ et en $x = \pi/2$ et change de convexité en $x = 0$. La fonction $\sin x$ semble donc satisfaire à nos besoins.

Soient en effet deux points dans le plan (x_1,y_1) et (x_2,y_2) et considérons la fonction $f(x) = a\sin[b(x + c)] + d$. Nous pouvons toujours déterminer les paramètres a, b, c et d (figure 4.8) de sorte que $f(x)$ admette une dérivée nulle en $x = x_1$ et en $x = x_2$ et change d'inflexion en un point $x^* = (x_1 + x_2)/2$. Il suffit de prendre

$$
\begin{aligned}
a &= \frac{1}{2}(y_2 - y_1), \\
b &= \frac{\pi}{x_2 - x_1}, \\
c &= -\frac{1}{2}(x_2 + x_1), \\
d &= \frac{1}{2}(y_1 + y_2).
\end{aligned}
$$

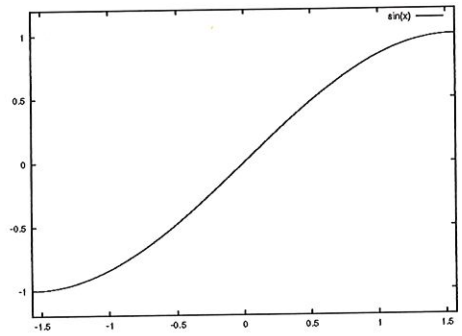


FIG. 4.7 – *La fonction* $\sin x$ *dans le plan.*

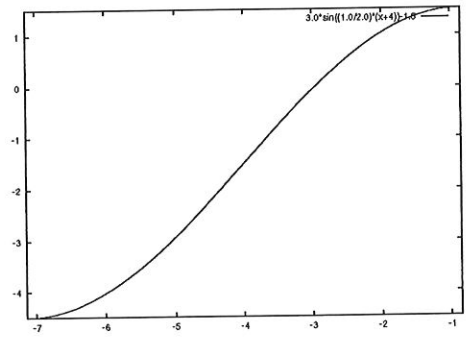


FIG. 4.8 – *Détermination des paramètres* a, b, c *et* d *de la fonction* $f(x) = a\sin[b(x+c)] + d$.

Nous désirons maintenant tracer un telle courbe entre deux points (notons les A et B) quelconques non plus dans le plan, mais dans l'espace. Quels

que soient ces deux points, nous pouvons trouver un plan perpendiculaire au plan XZ contenant ces deux points. Si ce plan est le plan XY ou un plan parallèle au plan XY, les formules ci-dessus sont valables. Sinon, nous pouvons appliquer une simple rotation d'axe Y à ce plan de sorte qu'il devienne paralèlle au plan XY. Une fois la courbe déterminée dans ce plan parallèle au plan XY, nous revenons par la rotation inverse dans le plan de départ.

Nous devons donc déterminer l'angle de rotation autour de l'axe Y. Nous suivons les étapes suivantes pour calculer cette valeur :

1. déterminer le plan passant par le point A (on aurait pu prendre B) et de vecteur normal (0,1,0) (ce plan est donc parallèle au plan XZ);
2. déterminer la projection B' du point B sur ce plan;
3. déterminer le vecteur $A - B'$ et le normer. Ce vecteur est à la fois dans un plan parallèle au plan XZ et dans le plan perpendiculaire au plan XZ, passant par les deux points;
4. l'angle que nous cherchons est l'angle formé par le vecteur e_1 et ce vecteur.

Remarquons que tous les outils nécessaires au calcul de cet angle (équation de plan, projection, ...) ont été présentés auparavant.

Maintenant que nous savons comment tracer une courbe entre deux points, nous pouvons tracer toutes les courbes de niveaux. Supposons alors que nous ayons deux faces dans l'espace; si nous dessinons les courbes de niveaux en joignant les côtés des faces qui se correspondent , nous obtenons un paysage qui est encore trop géométrique.

L'amélioration apportée est la suivante. La figure 4.11 fait apparaître quatre quarts d'ellipse. Alors, au lieu de joindre les points des côtés des faces jusqu'au sommet (provoquant des arêtes saillantes au niveau des droites joignant deux à deux les sommets), nous joignons les sommets aux points situés sur ces quarts d'ellipse comme montré à la figure 4.9.

Nous devons pour cela déterminer l'équation de chacun des quarts d'ellipse. Nous savons qu'une ellipse dans le plan XZ, centrée en $C = (cx,cz)$ et dont les axes sont parallèles aux axes X et Z a pour équation

$$\frac{(x - cx)^2}{a^2} + \frac{(z - cz)^2}{b^2} = 1,$$

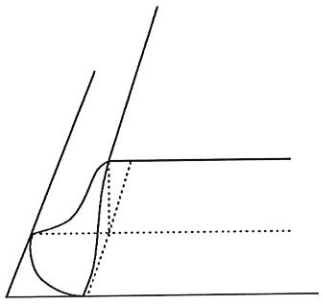


FIG. 4.9 – *Amélioration apportée pour réduire les angles.*

où (cx,cz) sont les coordonnées du centre de l'ellipse, a est la longeur du demi-grand axe et b la longueur du demi-petit axe (figure 4.10).

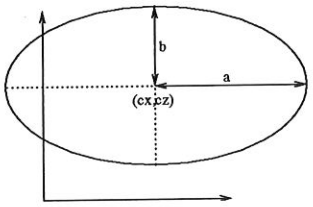


FIG. 4.10 – *Equation de l'ellipse dans le plan.*

Il est donc facile de déterminer l'équation de l'ellipse dans le plan XZ, si nous connaissons le centre de l'ellipse, deux de ses sommets (notons les $(s1x,s1z)$ et $(s2x,s2z)$) et si nous avons que l'ellipse a ses axes parallèles aux axes X et Z. En effet, dans ce cas $a = |s1x - s2x|$ et $b = |s1z - s2z|$.

Dès lors, pour déterminer l'équation de ces quarts d'ellipse, nous procédons comme suit (on s'appuiera sur la figure 4.11 pour comprendre les différents étapes).

1. nous calculons le centre de l'ellipse. Celui-ci vaut la projection du sommet sur le plan contenant la face "base";
2. nous calculons les deux sommets de l'ellipse. Pour déterminer ceux-ci, nous pouvons soit calculer la projection du centre de l'ellipse sur les

deux droites contenant les côtés de la face "base" ou calculer directement la projection du sommet sur ces deux droites;

3. nous effectuons alors des rotations sur les trois points que l'on vient de déterminer de sorte que l'ellipse dont on cherche l'équation soit dans un plan parallèle au plan XZ et ait ses axes parallèles aux axes X et Z. Ces rotations sont bien évidemment connues : il s'agit des trois composantes de l'objet *Angle* contenus dans le champ d'information de l'objet *CFace*;
4. nous pouvons alors déterminer les valeurs de cx, cz, a et b et nous obtenons l'équation de l'ellipse[18].
5. nous revenons dans le plan de départ en effectuant les rotations inverses.

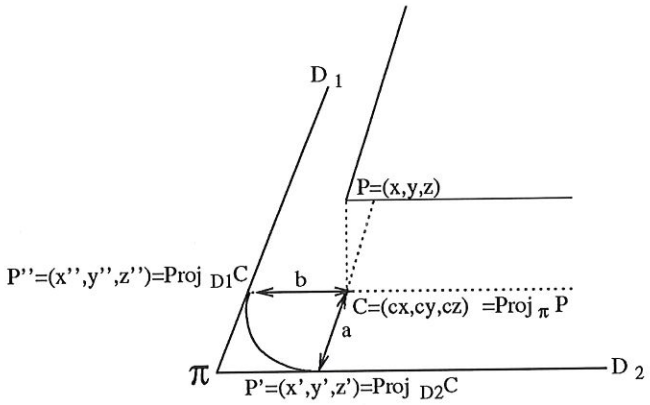


FIG. 4.11 – *Définition d'ellipses afin d'éliminer les arêtes.*

Remarque : la composante y des points n'intervient pas dans les calculs et donc, nous pouvons l'omettre.

L'utilisation des sinus et des ellipses pour tracer les courbes de niveau réduit considérablement l'aspect anguleux du paysage.

Lorsque nous arrivons en fin de structure, nous devons dessiner le sommet des plateaux. Nous avons alors le choix entre des sommets plats, en dôme ou dentelés. Pour construire le sommet en dôme, nous reprenons le même

18. Pratiquement, on détermine un ensemble de points se trouvant sur ce quart d'ellipse et c'est à ces points que l'on applique les rotations inverses de l'étape suivante.

principe de génération de courbes que ci-dessus, à la différence près que la petite face est réduite à un point (figure 4.12).

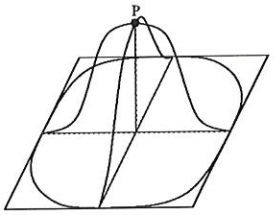


FIG. 4.12 – *Génération d'un dôme comme sommet d'un plateau.*

Utilisation d'une grille d'altitude

Nous nous posons alors la question de savoir comment à partir de cette liste de points générer une surface continue exprimant le relief. Comme nous le verrons plus loin, nous n'avons pas trop à nous soucier de ce problème: le langage *VRML* va nous simplifier la vie. Disons simplement ceci: il est possible en *VRML* de spécifier les dimensions d'une grille représentant un terrain et de spécifier l'altitude du terrain en chaque point de la grille. Notre travail se limite dès lors à la projection des points de la liste (les courbes de niveau) dans ce type de grille. Nous avons dans cette optique défini[19] un objet *C++* appelé *CGrille* contenant:

- *NbX* (*NbZ*): nombre de lignes (colones) dans la grille;
- *Esp*: espace entre les lignes et les colones. Nous imposons donc que l'espace entre les lignes et les colones soit égal;
- *Xmin* et *Xmin* (*Zmin* et *Zmax*): valeurs telles que les quatre coins de la grille sont situés en ($Xmin$,$Zmin$), ($Xmax$,$Zmin$), ($Xmax$,$Zmax$) et ($Xmin$,$Zmax$);
- *Altitude*: pointeur sur un tableau de réels. L'élément $j * NbX + i$ du tableau vaut l'altitude du noeud se trouvant à l'intersection de la ligne i et de la colone j. Ce tableau admet donc $NbX * NbZ$ éléments;
- *Objet*: pointeur sur un tableau d'entiers. L'élément $j * NbX + i$ détermine le type du noeud se trouvant à l'intersection de la ligne i et de la colone j (voir la section 4.3).

19. La définition de cet objet a été fortement influencée par l'outil *VRML* que nous utiliserons plus tard.

CGrille admet également les méthodes suivantes :

- *Init* : détermine et initialise les paramètres de la grille;
- *CalcAltitude* : cette méthode calcule les valeurs du tableau de réels *Altitude*. Le principe est le suivant : nous avons défini une grille constituée d'un certain nombre de lignes et de colones, ces lignes et colones étant séparées par une certaine distance. Nous appelons noeud le croisement entre une ligne et une colone. Nous associons chaque point de la liste au noeud le plus proche dans la grille (principe de la grille magnétique). La hauteur associée alors à un noeud atteint est la moyenne des hauteurs des points les plus proches du noeud.

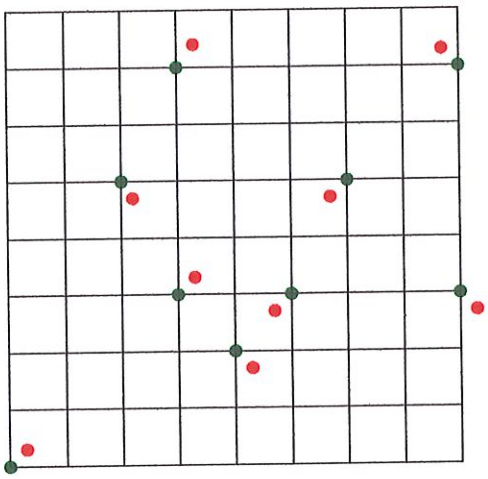


FIG. 4.13 – *Grille magnétique.*

Remarques : on peut se poser la question de savoir si tous les noeuds de la grille sont atteints par les points de la liste lors de la projection. En début de programme, nous définissons les paramètres (nombre de lignes et de colones ainsi que l'espace entre ces lignes et ces colones) de la grille (ces paramètres sont spécifiés par l'utilisateur comme nous le verrons au chapitre 5). Nous ajustons alors en fonction de ces paramètres la génération des points constituant les courbes de niveau de sorte que tous les points de la grille soient atteints. Remarquons également que le terrain que l'on obtient en utilisant cette méthode n'est pas tout à fait le terrain dont on a généré les courbes de niveau.

4.2.4 Récapitulatif des paramètres importants pour l'interprétation du paysage

Ces paramètres sont au nombre de trois. Nous allons prendre un petit exemple qui va nous permettre de mieux saisir l'importance de ces paramètres.

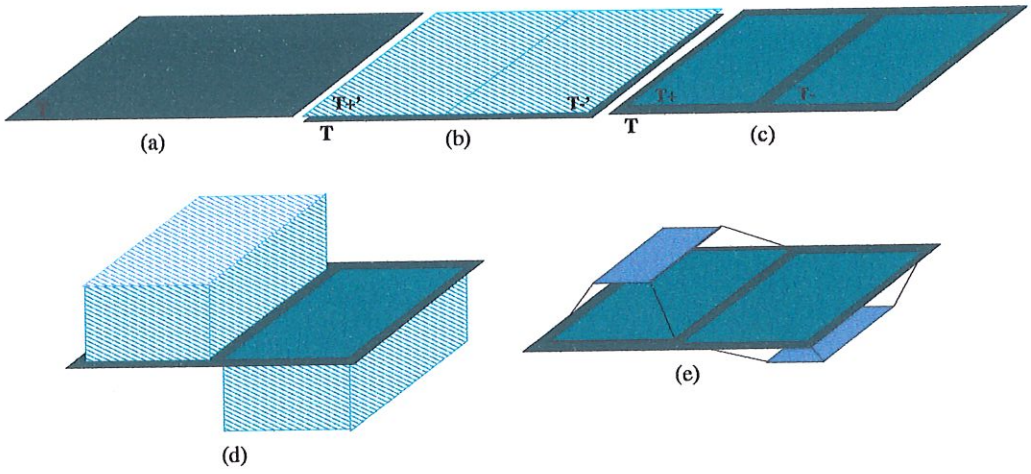


FIG. 4.14 – *Importance des paramètres* `ESP_BANDE`, `ESP_PENTE` *et* `ALT`.

Suppsons que l'on désire représenter un disjonction.

Tout d'abord, on se définit une face dans l'espace que l'on note T comme indiqué à la figure 4.14(a).

Comme il s'agit d'une disjonction, nous devons ensuite créer deux faces (T_+ et T_-). Pour créer ces faces, nous partageons en deux la face initiale. Nous obtenons ainsi deux faces notées T'_+ et T'_- (figure 4.14(b)). Si nous laissons ces deux faces comme cela, nous n'avons aucune séparation (aucun "no man's land") entre les deux modalités. Nous effectuons donc sur ces deux faces une contraction d'un certain facteur afin d'obtenir T_+ et T_-. Nous appelons ce facteur `ESP_BANDE` : c'est lui qui détermine (comme son nom le laisse sous-entendre) la largeur des bandes entre les structures. Dans le programme, `ESP_BANDE` est choisi aléatoirement entre 0.9 et 0.95 (nous ne mettons pas 1 pour être sûr d'avoir toujours une bande).

Il nous faut encore déterminer les sommets des plateaux. Nous reprenons pour cela une copie de T_+ et T_-. Nous translatons alors ces copies vers le haut (valorisation positive) ou vers le bas (valorisation négative) (figure 4.14(c)). La hauteur dont nous translatons ces deux faces constitue un autre paramètre qu'on a appelé `ALT`. Plus nous désirons une dénivellation importante et plus `ALT` doit être grand; inversément, plus `ALT` est faible et plus les dénivellations sont douces.

Le dernier paramètre sur lequel nous pouvons jouer s'appelle `ESP_PENTE`. En effet, translater ces deux copies ne suffit pas : si nous ne contractons pas ces deux copies, les pentes seront verticales (figure 4.14(d)). Ce nouveau facteur de contraction s'appelle `ESP_PENTE`. Dans le programmme, il est choisi aléatoirement entre 0.6 et 0.7. En modifiant `ALT` et `ESP_PENTE`, nous pouvons ajuster une pente moyenne (figure 4.14(e)) qui nous semble refléter le mieux les relations qu'il y a entre les modalités de la disjonction[20].

Remarque : il faut noter que `ALT` constitue en fait un rapport. Pour être exact, `ALT` vaut la pente moyenne des dénivelés. Ce choix se justifie pour les raisons suivantes. On sait que la pente d'une droite vaut "ce dont je monte sur ce dont j'avance". Or, nous ne connaissons pas à l'avance précisément de combien nous voulons monter (on ne connait pas à l'avance la hauteur relative des plateaux). Par contre, on peut se décider entre des pentes raides ou des pentes douces.

4.3 Génération du décor

Une fois que le relief[21] a été généré, nous pouvons "habiller" ce relief et créer ainsi un paysage de plus en plus réaliste. Tous les objets que l'on introduit dans le paysage sont des objets définis en *VRML*. Cependant, dans cette section, nous nous attardons davantage à expliquer comment ces objets sont insérés dans le paysage. Nous réservons pour l'annexe A les détails concernant la création de ces objets *VRML*.

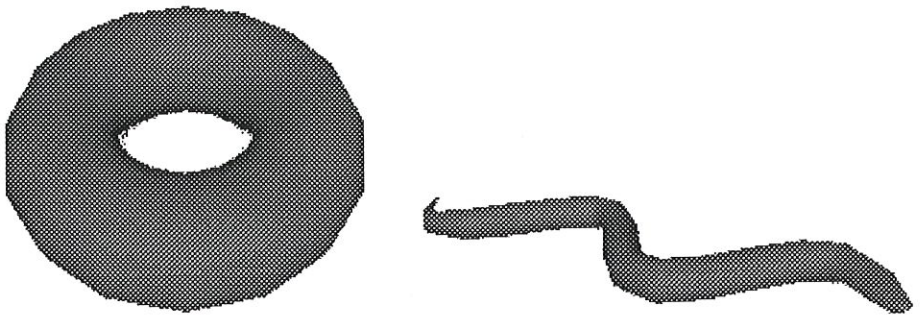
20. Une pente raide traduit le fait qu'il est très difficile de passer d'une modalité à l'autre; une pente douce rend compte de la différence entre les deux modalités mais indique une relative aisance pour passer de l'une à l'autre.

21. Nous entendons désormais par relief la grille et les altitudes associées à chaque noeud.

4.3.3 Les objets linéaires

Pour placer des objets linéaires dans le paysage, nous devons connaître toute une série de points par lesquels passe cet objet. C'est le cas si nous désirons introduire une route, une rivière, un ligne haute tension, ... dans le paysage. En fait, ce type d'objet est "*dessiné à même le paysage*", par opposition aux objets ponctuels que l'on insérait dans le paysage . En effet, une fois les points par lesquels passent l'objet déterminés, nous pouvons interpoler la trajectoire et la dessiner. Cette opération est d'autant plus facile qu'on peut définir en *VRML* des formes dites extrudées. Ce type de formes est généré de la façon suivante. Nous définissons premièrement une courbe dans l'espace appelée **épine dorsale** ou **génératrice**. Ensuite, on se donne une face quelconque dans le plan (un rectangle, un rond, un hexagone, ...) appelée **coupe transversale**, **tranche** ou encore **section**. La forme finale est le solide obtenu en déplaçant la tranche le long de la courbe génératrice (voir l'annexe A pour de plus amples renseignements).

Exemples : nous pouvons construire un tore en prenant comme épine dorsale un cercle et comme coupe transversale un disque.



(a) Le tore. (b) Forme plus compliquée.

FIG. 4.15 – *Différents solides obtenus avec des formes extrudées.*

Ainsi, les points par lesquels passe par exemple une route nous permettent de définir une courbe qui servira de génératrice. Nous choisissons alors comme

section un rectangle de longeur et de largeur correspondant à la largeur et à l'épaisseur de la route.

Remarques : on peut obtenir des formes plus compliquées par ce procédé en définissant pour chaque point de la génératrice une orientation ainsi qu'un facteur d'échelle à appliquer à la section en ce point.

4.3.4 L'ambiance

En jouant avec la lumière (position de la source lumineuse, couleur de la lumière ambiante et luminosité), avec la couleur du ciel et avec la texture du sol, nous pouvons générer différentes ambiances pour le paysage.

Ainsi, la source de lumière sera positionnée loin à l'horizon côté est (respectivement côté ouest) si nous désirons créer une ambiance de lever (respectivement de coucher) de soleil . Elle sera de couleur légèrement orangée (respectivement bleutée) et la luminosité sera faible. Pour simuler l'ambiance en plein midi, la source de lumière sera placée au zénith du paysage; la luminosité sera abondante et la lumière sera de couleur blanche.

La couleur du ciel a aussi un rôle important : on la choisira dans les tons oranges au lever du soleil, dans les tons bleus au plein midi, dans les tons mauves au coucher du soleil, dans les tons gris si nous désirons simuler un ciel hivernal, ...

La couleur du sol a également son importance. Ainsi, dans le programme nous avons prédéfini quatre types de couleur rendant compte de quatres sortes de paysages que l'on peut rencontrer dans la nature : vert pour les paysages de type rural, blanc pour les paysages enneigés, orange pour les paysages désertiques et gris pour les paysages de montagne (caillasse).

Nous n'allons pas énumérer ici toutes les ambiances réalisables en jouant avec ces paramètres. Nous signalons toutefois que pour le moment, elles sont prédéfinies dans le programme (voir chapitre 5). Néanmoins, il est envisageable de modifier le programme de sorte que le chercheur puisse lui-même définir ces paramètres et ainsi créer ses propres ambiances. Aussi, en parcourant l'annexe A, le lecteur pourra se rendre compte des immenses possibiltés qu'offre le *VRML* afin d'introduire une ambiance dans une scène.

4.4 Conversion du paysage au format *VRML*

Nous avons à présent un aperçu du fonctionnement du programme. Nous avons vu comment nous nous y prenions pour générer un relief exprimant les résultats d'une analyse structurale. Il reste cependant un petit point à éclaircir : la visualisation des scènes. En effet, nous avons généré en fin de compte une grille; à chaque noeud de celle-ci, nous avons associé une altitude si bien que cette grille exprime le relief. A ce relief, nous avons ajouté des objets et une ambiance. Il nous reste à voir comment dans le programme nous faisons le lien entre ces résultats et les fichiers *VRML* visualisables avec un navigateur adéquat (tous ces termes vous sembleront moins barbares après la lecture de l'annexe A). Il nous reste donc à aborder la conversion de l'objet *CGrille* en un objet (un fichier .wrl) *VRML*.

Pour rappel, un objet de type *CGrille* contient les champs suivants[24] :

- *Esp*,
- *NbX*, *NbZ*,
- *Xmin*, *Xmax*, *Zmin*, *Zmax*,
- *Altitude*,
- *Objet*.

Nous allons anticiper quelque peu l'annexe A, car nous allons présenter ici l'objet *VRML* permettant de générer des terrains. Dans un soucis de simplicité et parce que nous désirons aller à l'essentiel, seules les propriétés de cet objet dont nous nous sommes servis seront décrites. Pour plus de détails, on se réfererra à l'annexe A ou à [2].

L'objet *VRML* permettant de créer un terrain cahoteux s'appelle **ElevationGrid**. Le principe de base est le suivant : nous définissons une grille (le nombre de lignes et de colones et l'espace entre ces lignes et ces colones sont sous notre contrôle) et nous spécifions alors l'altitude de chaque point de la grille. Le navigateur *VRML* construit lui-même les faces pour chaque carré

24. Nous ne faisons que les énumérer. Reporter-vous à la section 4.2.3 pour de plus amples informations.

de la grille afin de générer notre terrain.

La syntaxe (élémentaire) de ce bloc est la suivante[25] :

TAB. 4.1 – **Syntaxe du bloc ElevationGrid**

ElevationGrid {			
xDimension	0	field	SFInt32
zDimension	0	field	SFInt32
xSpacing	0.0	field	SFFloat
zSpacing	0.0	field	SFFloat
height	[]	field	MFFloat
color	NULL	exposedField	SFNode
colorPerVertex	TRUE	field	SFBool
creaseAngle	0	field	SFFloat
}			

Remarques : la première colone désigne le nom des champs du noeud, la deuxième colone leur valeur par défaut, la troisième leur type et la dernière le type de valeur qu'ils peuvent prendre.

Les valeurs de **xDimension** et **zDimension** spécifient le nombre de points dans la grille dans la direction X et Z. Il est évident que pour construire une grille, ces valeurs doivent être supérieures à un. Les valeurs par défaut 0 indiquent que aucune grille n'est construite.

Les valeurs de **xSpacing** et **zSpacing** spécifient la distance entre les lignes et les colones dans la grille.

Le champ **height** spécifie la liste des altitudes, une pour chaque point de la grille. Les altitudes sont listées par lignes (c'est à dire dans le sens de l'axe X) : ainsi, l'élément $j * xDimension + i$ de la liste est l'altitue du point (i,j) de la grille. Il est évident que le nombre de valeurs contenues dans cette liste doit valoir $xDimension * zDimension$, i.e. le nombre de points de la grille.

Pour définir la couleur du terrain, il existe deux façons de procéder : soit nous définisons une couleur globale pour le terrain à l'aide du noeud **Ap-**

25. Pour plus de détails, reportez-vous à l'annexe A.

pearance du noeud **Shape**[26], soit nous définissons une couleur par face. Ceci permettra par exemple d'obtenir des variations de couleur en fonction de l'altitude ou en fonction de la position (ainsi, dans le programme, lorsqu'on s'approche des limites du paysage, le sol devient noir). Nous pouvons réaliser cela à l'aide des champs **colorPerVertex** et **color**. **colorPerVertex** vaut **FALSE** si nous voulons assigner une couleur à chaque carré de la grille (pour réaliser un damier par exemple) et vaut **TRUE** si nous désirons assigner une couleur à chaque point de la grille. Dans ce cas, les couleurs varient régulièrement dans chaque face (interpolation des couleurs). Dans le programme, **colorPerVertex** vaut **TRUE**. **color** est un noeud de type **Color**. La syntaxe de ce noeud est très simple :

TAB. 4.2 – **Syntaxe du bloc Color**

```
Color {
  color   []   exposedField   MFColor
}
```

Le champ[27] **color** du noeud **Color** spécifie (dans notre cas) la liste des couleurs[28] de chaque point (ou face) de la grille.

creaseAngle est un réel permettant d'adoucir les angles entre les faces générées pour construire le terrain. **creaseAngle** spécifie en fait la valeur d'un angle en radian tel que si l'angle entre deux faces adjacentes est plus petit que cette valeur, la jonction entre ces deux faces sera légèrement ombrées de façon à "masquer" les angles.

Finalement, il faut savoir que le tout premier point de la grille est situé au point (0 0 0) et que donc la grille se situe dans le quadrant des x et des z positifs.

Nous en savons suffisamment pour pouvoir expliquer comment à partir de l'objet *CGrille*, nous créons un noeud de type **ElevationGrid**. Les valeurs des champs **xDimension** et **zDimension** sont données par les valeurs

26. Retenez pour le moment que le noeud **Shape** englobe le noeud **ElevationGrid**.

27. Attention de ne pas confondre les deux champs **color**, l'un étant un noeud et l'autre une liste de triplets de réels.

28. On définit en *VRML* une couleur par un triplet de réels, tous compris entre 0 et 1, chacun d'eux donnant les composantes rouge, verte et bleue de la couleur.

des champs *xDim* et *zDim* de l'objet *CGrille*. **xSpacing** et **zSpacing** ont la même valeur : celle-ci est donnée par le champ *Esp*. La liste **height** des altitudes des points de la grille est donnée par le tableau *Altitude*: l'ordre dans lequel sont listées les hauteurs est en effet le même. Finalement, les autres champs du noeud **ElevationGrid** sont déterminés indépendammment de l'objet *CGrille*. Ainsi, la couleur du sol dépend du type de texture choisi et la valeur de **creaseAngle** est fixée arbitrairement à π de sorte que tous les angles entre toutes les faces paraîtront moins saillant. Enfin, *Xmin*, *Xmax*, *Zmin* et *Zmax* vont nous permettre de centrer la grille à l'origine.

Le lecteur a désormais une idée générale du principe de fonctionnement du programme. Comme nous le constatons, beaucoup d'outils ont été mis en place pour atteindre le résultat présenté au chapitre 2. Les bases (au moins quelques bonnes idées) du programme qui génèrera à l'avenir toutes les sortes de structures rencontrées en analyse structurale semblent ainsi avoir été posées. Le travail est certes encore long mais nous allons présenter dans les dernières pages de ce mémoire quelques résultats qui paraissent encourageant pour la suite du projet.

Chapitre 5

Utilisation du programme

Ce chapitre est destiné à expliquer comment installer le programme et comment s'en servir. Nous présentons également une visionneuse particulière provenant de chez *Parallel Graphics*, utilisée dans le cadre de ce travail.

Remarque: le programme ainsi que la visionneuse ne tournent que sur des plates-formes *Windows*.

5.1 Installation du programme et création des répertoires nécessaires

La première étape dans l'installation du programme est la création d'un répertoire dans lequel nous copions le document **Mémoire.zip**.

Une fois ce document décompressé, le répertoire courant (on le note ./) présente la structure suivante:

- ▼ [Programme]
  - ▼ [Exécutable]
    - ▶ [Fichiers .obj]
  - ▶ [Sources]
- ▼ [Analyse]
  - ▶ [Lien]
- ▼ [Résulats]
  - ▶ [Paysage]

- ▶ [Panneau]
- ▶ [Relief]
- ▶ [Ambiance]
- ▼ [Décor]
  - ▶ [Bibliothèque]

Nous allons détailler le contenu de chacun des ces répertoires.

- Le répertoire **Programme** contient tous les fichiers relatifs au programmme. Ainsi, dans le répertoire **Exécutable**, nous trouvons le programme principal, le fichier d'installation de la visionneuse et le répertoire **Fichiers .obj** contenant les fichiers `.obj` provenant de la compilation. Le répertoire **Sources** contient tous les fichiers sources du programme écrit en *C++*.
- **Analyse** contient les fichiers de données encodant l'analyse et respectant le format décrit à la section 4.2.2. Les fichiers vers lesquels nous avons des liens à partir du paysage sont stockés dans le répertoire **Lien**
- Tous les fichiers `.wrl` décrivant la scène finale se trouvent dans le répertoire **Résultats**. Afin de rendre faciles l'édition et la lecture des scènes, nous avons divisé **Résultats** en quatre répertoires.
  - Les paysages définitifs sont contenus dans le répertoire **Paysage**.
  - Le répertoire **Relief** contient les fichiers décrivant les terrains "bruts", c'est à dire auxquels nous n'avons pas encore ajouté de décor (objets et ambiance). Ces fichiers ne contiennent en fait que les blocs *VRML* **ElevationGrid**.
  - Les fichiers de description des panneaux que l'on insère dans le paysage se trouvent dans le répertoire **Panneau**.
  - Tous les fichiers ayant trait à l'ambiance (couleur du ciel, source de lumière, ...) imprégnant la scène se trouvent dans le répertoire **Ambiance**.
  - Enfin, le répertoire **Décor** est composé des fichiers relatifs à l'insertion d'objets dans le paysage. Les différents objets *VRML* prédéfinis sont stockés dans le répertoire **Bibliothèque**.

Remarque concernant le nom des fichiers: l'utilisateur introduit le nom qu'il veut pour son paysage final. Par exemple, il peut appeler celui-ci *analyse.wrl*. Tous les fichiers utiles à la description de la scène, générés par le programme ont alors un nom commençant par ce nom de fichier (moins

l'extension), suivi d'un nom spécifique. Dans notre exemple, le fichier contenant l'ambiance de la scène s'appellerait *analyseAmbiance.wrl*. Cela permet d'identifier les fichiers décrivant une même scène.

5.2 Télécharger et installer une visionneuse

Pour pouvoir visualiser les scènes, nous devons disposer d'un "*browser*" *VRML*. Comme nous l'avons déjà souligné, nous trouvons sur Internet un vaste choix de visionneuses. Nous trouverons d'ailleurs sur le site

`http://www.sdsc.edu/vrml`

une liste de tous les "*browsers* actuellement disponibles. Ce site constitue à l'heure actuelle le site de référence en ce qui concerne le *VRML*.

Dans le cadre de ce travail, nous avons choisi de travailler avec la visionneuse *Cortona* de chez *Parallel Graphics*. Il n'y a pas de raison particulière justifiant ce choix si ce n'est que nous trouvons qu'elle permet une navigation aisée dans la scène. Cette visionneuse est téléchargeable très facilement[1] à partir du site

`http://www.parallelgraphics.com/`

Afin d'éviter ces opérations, nous fournissons le fichier d'installation de ce "*browser*" *VRML*. Il suffit de cliquer deux fois sur l'icône

pour l'installer.

1. C'est réellement très facile de la télécharger. En effet, nous sommes habitués à devoir suivre des dédales quand nous désirons télécharger un programme à partir d'Internet, mais ici, ce n'est pas le cas.

5.3 Utiliser le programme

Pour lancer le programme, nous cliquons deux fois sur l'icône

Une petite page d'accueil présentant brièvement le programme apparait.

Si nous cliquons sur le bouton "*Continuer*", une fenêtre d'ouverture de fichier s'affiche à l'écran (figure 5.1).

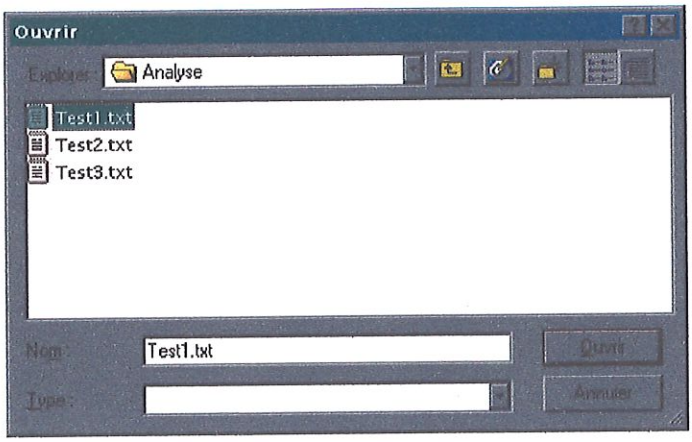


FIG. 5.1 – *Sélection d'un fichier.*

Nous choisissons alors un fichier contenant une analyse. Ce fichier doit respecter un certain format. Si une erreur survient dans la lecture du fichier, une fenêtre nous en avertit (figure 5.2).

Si la syntaxe du fichier est conforme, la fenêtre 5.3 apparait. Celle-ci nous permet de sélectionner divers paramètres propres à la génération du paysage comme la taille du paysage, le type d'ambiance, ou encore les objets insérés dans le paysage.

Si nous cliquons sur le bouton "*Autres Paramètres*", nous obtenons obtenons la figure 5.4.

FIG. 5.2 – *Erreur de lecture du fichier: un mot-clé a reçu une mauvaise valeur.*

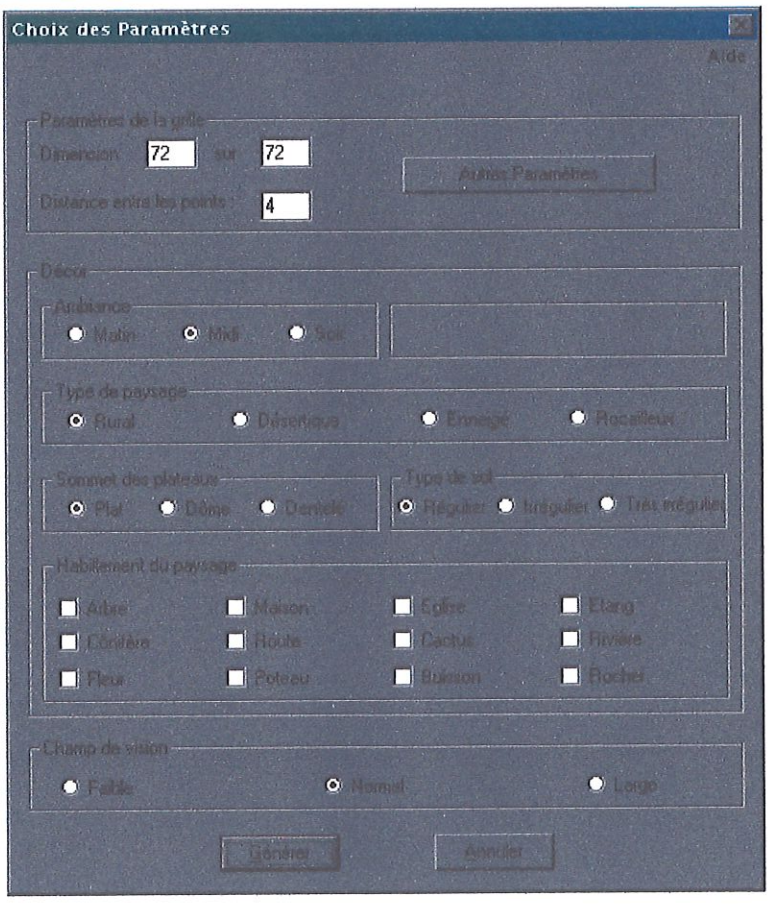


FIG. 5.3 – *Fenêtre principale pour la sélection des paramètres.*

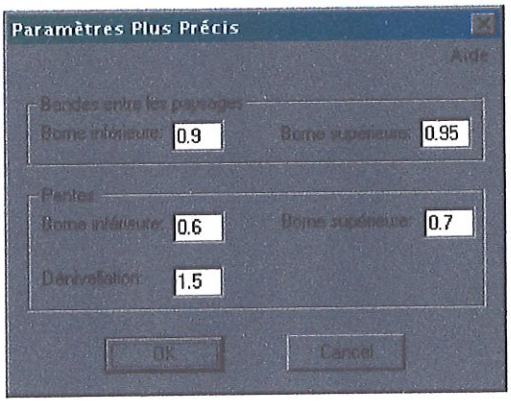


FIG. 5.4 – *Fenêtre secondaire pour la sélection des paramètres.*

Nous pouvons ici modifier les paramètres concernant la pente moyenne des dénivelés et l'espace entre les structures.

Nous terminons la sélection des paramètres en cliquant sur "*Générer*", ce qui lance la génération du paysage en fonction des options choisies.

Une fenêtre d'enregistrement de fichier apparait ensuite (figure 5.5). Celle-ci s'ouvre automatiquement dans le bon répertoire (c'est à dire le répertoire **Paysage**) et nous permet d'introduire le nom qu'aura le fichier `.wrl` contenant le paysage.

Une dernière fenêtre nous fait savoir quand l'exécution du programme est terminée (figure 5.6).

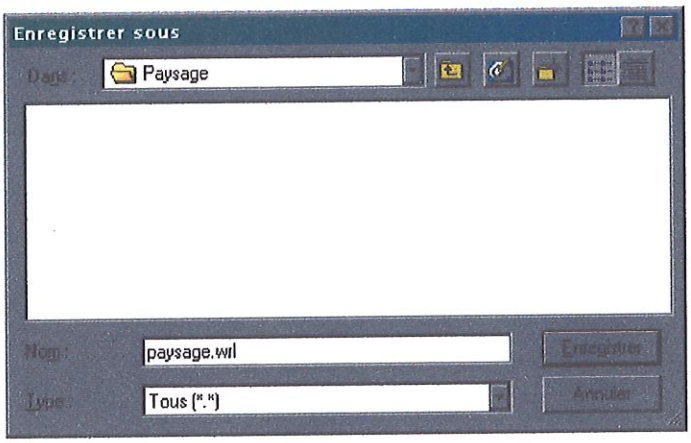


FIG. 5.5 – *Enregistrement d'un fichier.*

FIG. 5.6 – *Fin de l'exécution du programme.*

5.4 Naviguer dans les scènes

Une fois généré, nous désirons visionner le fichier `.wrl` contenant le paysage. Pour ce faire, il faut cliquer deux fois sur l'icone

ou sur tout autre fichier d'extension `.wrl` selon le nom donné au fichier. Cela provoque le lancement de *Internet Explorer* et après quelques instants, nous obtenons un page Internet affichant notre paysage.

Il existe plusieurs manières de se déplacer dans la scène. En effet, nous disposons de trois sortes de navigation: "*Fly*" (voler), "*Walk*" (marcher) et "*Study*" (examiner). A ces trois types de navigation, nous pouvons encore associer quatres sortes de mouvement: un mouvement dans le plan horizontal ("*Plan*"), un déplacement dans le plan vertical ("*Pan*") (sauf pour le type de navigation "*Walk*"), un mouvement de rotation de gauche à droite et de bas en haut[2] ("*Turn*") et un mouvement de rotation autour de l'axe correspondant à la direction dans laquelle observateur regarde ("*Roll*"). Nous utilisons les touches directionnelles pour nous déplacer une fois le type de mouvement sélectionné.

Nous pouvons également nous mouvoir dans le paysage en utilisant les différents points de vue. Il suffit d'utiliser les boutons "*View*" ou d'aller dans le menu de positionnement (cliquer avec le bouton droit de la souris pour le faire apparaitre) et de choisir un point de vue (figure 5.7).

Signalons encore qu'il est possible de spécifier sa vitesse de déplacement dans la scène à l'aide du menu contextuel obtenu en cliquant avec le bouton droit de la souris.

Enfin, lorsqu'il passe devant un panneau, le curseur de la souris change d'aspect (il se transforme en une petite ancre), ce qui signifie qu'il y a un lien à partir du panneau. Ainsi, si on clique sur le panneau, on est renvoyé

2. Et bien évidemment de droite à gauche et de haut en bas.

à ce lien. Nous constatons également dans la barre de statut (dans le coin inférieure gauche) l'affichage du texte contenu dans le champ *Expl* de l'objet *CArbre*.

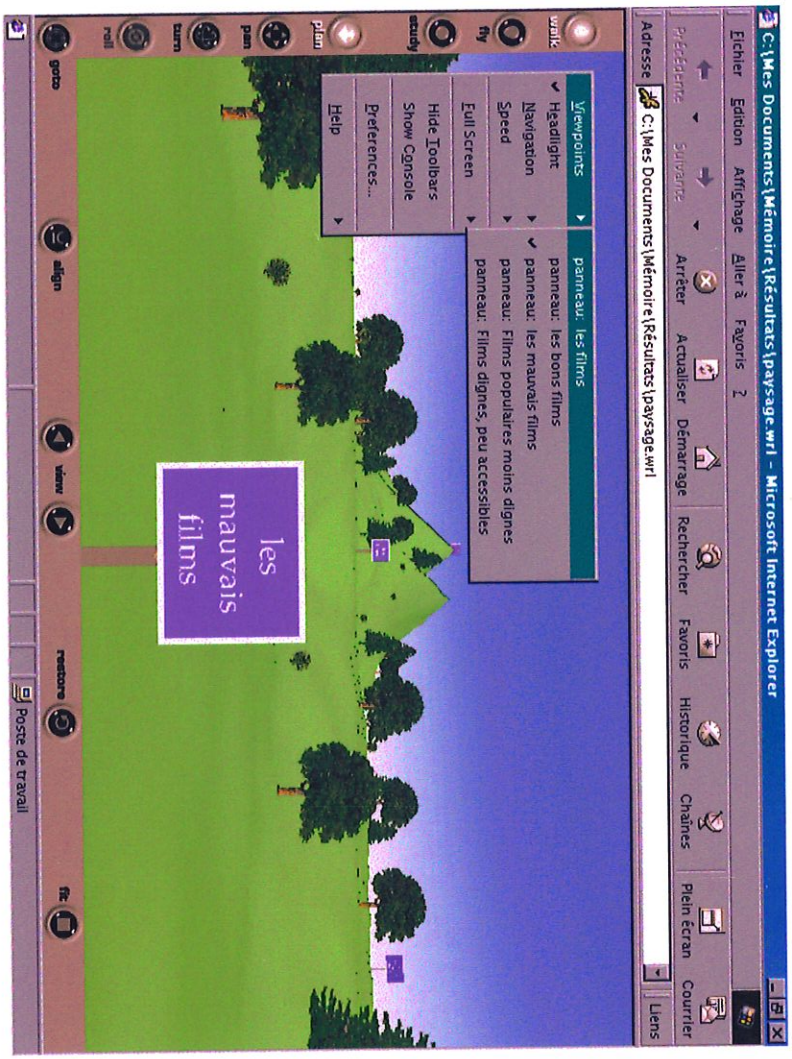


FIG. 5.7 – *Sélection d'un point de vue.*

5.5 Quelques résultats

Il est enfin temps de présenter quelques vues de paysages générés avec notre programme. Les fichiers sources de ces exemples sont livrés dans le répertoire **Analyse**.

Les différentes scènes que nous présentons ici, sont les représentations en paysage des trois exemples énoncés la première fois au chapitre 1 et desquels nous avons explicité les fichiers encodant leurs analyses au chapitre 4.

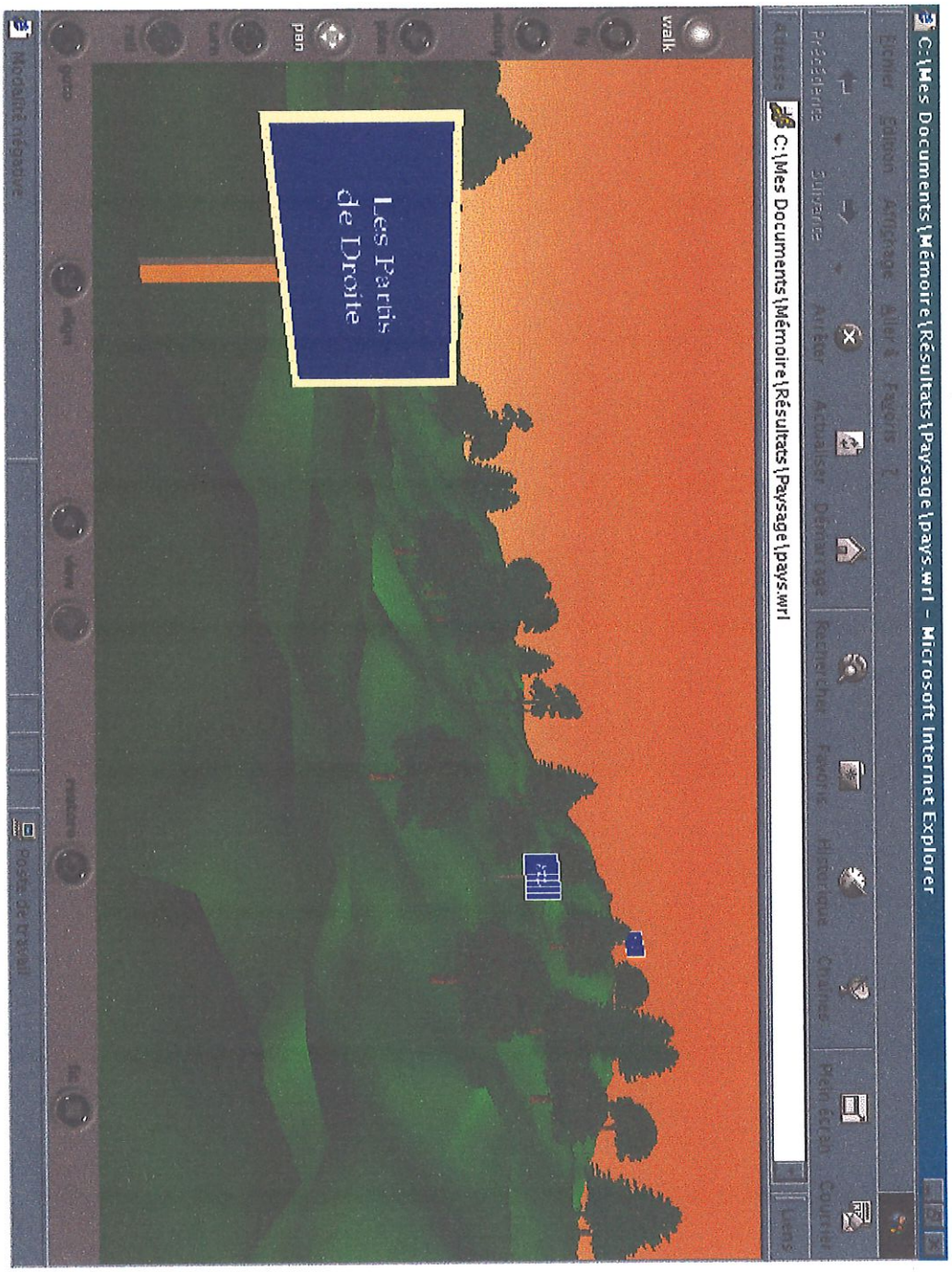


FIG. 5.8 – *Paysage rural, le matin, sommet des plateaux en dôme, sol très irrégulier, ajout d'arbres, de buissons et de cônifères.*

FIG. 5.9 – *Paysage rural, le soir, sommet des plateaux dentelés, sol régulier, ajout de maisons, d'une église et d'arbres.*

FIG. 5.10 – *Paysage enneigé, à midi, sol irrégulier, ajout d'un étang et de cônifères.*

Conclusion

Nous concluons ce travail en faisant le point sur ce qui a été fait et en proposant des idées susceptibles d'une part d'améliorer le fonctionnement du programme et d'autre part de rendre les réalités virtuelles obtenues encore plus proches de ce que peut en attendre le chercheur.

Tout d'abord, nous tenons à souligner que le travail est encore long jusqu'à la mise au point du logiciel tel que définit dans le cahier des charges.

Ainsi, le programme réalisé dans le cadre de ce travail ne génère que les paysages sur base des résultats de l'analyse. Nous n'avons pas du tout tenu compte des autres contraintes spécifiées dans le cahier des charges telles que l'interface d'encodage des données[3]. Nous n'avons pas non plus envisager le traitement des cas particuliers présentés à la section 2.4.4, tels que le dilemme, les réalités exclues, ...

Pour le moment, nous n'introduisons une composante aléatoire dans la génération du paysage qu'au niveau des dimensions des différentes faces. On réduirait certainement encore l'aspect géométrique du paysage en situant et en orientant aléatoirement ces faces dans l'espace. Notons cependant qu'il devra tout de même y avoir des bornes sur les déplacements et sur les rotations etque pourront effectuer ces faces afin que la "charpente" du paysage (définie par ces faces) reste cohérente. En quelque sorte, ces mouvements aléatoires auront lieu autour d'une "position d'équilibre" et ne pourront dépasser une certaine amplitude. Tous les outils permettant de réaliser cela sont implémentés (rotation des faces autour d'un axe quelconque, translation d'une face, ...) et seul le manque de temps nous a empêché de "mettre en musique"

3. Nous avons certes implémenté un petit interface mais il n'a rien de définitif. Il nous permet simplement d'introduire de façon conviviale les paramètres du programme.

l'introduction de ces différentes composantes aléatoires dans la génération du paysage.

De même, pour le moment le chercheur sait introduire divers paramètres (taille des plateaux, pourcentage des pentes, espace entre les plateaux, ...) mais ces paramètres se répètent pour tout le paysage. On devrait pouvoir définir ces paramètres pour chaque niveau dans la structure.

Nous avons choisi la fonction *sinus* et utilisé des ellipses pour générer les courbes de niveau et les résultats obtenus sont encourageants. Néanmoins, il pourrait être intéressant de déterminer d'autre type de courbes vérifiant la propriété d'admettre une dérivée nulle en deux points de l'espace et un seul point d'inflexion situé entre ces deux points. En combinant plusieurs courbes de ce type, on accentuerait davantage le réalisme dans la forme du relief.

Nous avons vu également que pour générer les courbes de niveau, nous calculons toute une série de points que nous projetons ensuite dans une grille. A l'évidence, nous générons beaucoup trop de points par rapport au nombre de noeud dans la grille et ce afin que tous les noeuds de la grille soient atteints lors de la projection. Nous n'avons pas trouvé de solution immédiate à ce problème. La question reste donc ouverte.

Finalement, nous avons eu l'idée d'utiliser le langage *VRML* peut-être tardivement. Nous n'avons donc pas exploité toutes les possibilités qu'offre cet outil. Tout d'abord, nous introduisons les objets dans le paysage de façon aléatoire. Ce ne devrait pas être trop difficile d'écrire un petit programme qui permette au chercheur d'introduire là où il veut dans le paysage des objets via une méthode de type *glisser-lâcher*. En effet, le *VRML* permet de réaliser de telles actions. De plus, pour l'instant les paysages générés sont dépourvus d'animation. On pourrait envisager d'en inclure. De même, rien ne devrait nous empêcher d'introduire des sons dans le paysage, que ce soit des chants d'oiseaux et le bruit du vent ou bien la lecture du texte ayant généré l'analyse. A la place d'écrire des mots dans le paysage aux endroits appropriés, on pourrait inclure des bandes sonores de sorte que lorsqu'on s'approche de ces endroits, on déclenche la lecture de celles-ci. Nous n'avons pas non plus utilisé tous les objets *VRML* permettant de créer des ambiances particulières. Ainsi, nous pourrions ajouter du brouillard dans le paysage.

Nous pourrions continuer longtemps l'énumération des petites touches à apporter au paysage pour accroître son réalisme et permettre au chercheur de traduire sa sensibilité quant à l'analyse retenue mais nous préférons laisser le choix de ces dernières à d'autres car elles sont tributaires de l'imagination de chacun.

Bibliographie

[1] A. Aho and J. Ullman. *Concepts fondamentaux de l'algorithmique*, pages 236–244. Dunod, 1992.

[2] A. L. Ames, D. R. Nadeau, and J. L. Moreland. *VRML 2.0 Source Book.* John Wiley, 1997.

[3] Stuart K. Card, Jock D. Mackinlay, and Ben Schneiderman. *Readings in Information Visualisation, Using Vision to Think*, pages 1–35. Morgan Kaufmann, 1999.

[4] P.A. Egerton and W.S. Hall. *Computer Graphics, Mathematical First Steps.* Prentice Hall, 1999.

[5] Joël Grad and Stéphane Grad. *Génération de vues 3D de paysages.* Facultés Universitaires Notre Dame de la Paix, Namur, 1999.

[6] A.J. Greimas. *Sémantique Structurale.* Librairie Larousse, 1966.

[7] J.P. Hiernaux. *L'institution culturelle, Méthodes de description structurale.* Presses Universitaires de Louvain (U.C.L), 1977.

[8] Alexandre Patrick. Evoq, projet de création d'un logiciel d'analyse de contenu (projet fsr). Cahier des charges.

[9] Anne Piret, Claude Florkin, Luc Neuberg, and Anne Wallemacq. Drawing cognitive landscapes, exploring the potentialities of structural(ing?) analysis. *Paper presented at the 6th Workshop on Managerial and Organizational Cognition: "Rethinkink in/by Organization/Collectivities", University of Essex, Colchester, U.K., June 1-3, 1999.*

[10] Anne Piret, Jean Nizet, and Etienne Bourgeois. *L'analyse structurale, une méthode d'analyse de contenu pour les sciences humaines*, pages 13–90. De Boeck & Larcier, 1996.